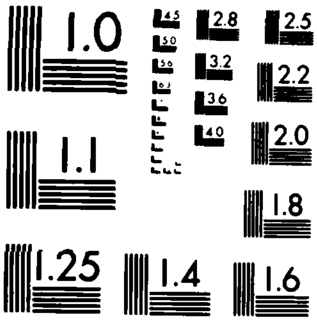
AD-A152 166   IMPULSE: A CODE FOR HIGH-RESOLUTION IMPULSE RESPONSE   1/1
FUNCTIONS IN RANGE-DEPENDENT DUCTS(U) NAVAL OCEAN
SYSTEMS CENTER SAN DIEGO CA  R C SHOCKLEY ET AL. DEC 84
UNCLASSIFIED   NOSC/TR-995   F/G 17/1   NL

END

FILMED



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

**Technical Report 995**

December 1984

Final Report

October 1983 — August 1984

# IMPULSE: A CODE FOR HIGH-RESOLUTION IMPULSE RESPONSE FUNCTIONS IN RANGE-DEPENDENT DUCTS

R. C. Shockley
J. Northrop

Prepared for
Naval Material Command
Independent Research Programs

**Naval Ocean Systems Center**
San Diego, California 92152

Approved for public release; distribution unlimited.

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

F. M. PESTORIUS, CAPT, USN
Commander

R.M. HILLYER
Technical Director

## ADMINISTRATIVE INFORMATION

This task was performed for Naval Material Command, Independent Research Programs, Washington, DC 20360, under program element 61152N, subproject ZR0000101. The work was carried out under the NOSC Independent Research project on Linear and Nonlinear waves.

This report presents a new computer algorithm for identifying multipath arrivals in underwater sound channels and other ducts.

Released by
R. W. Larsen, Head
Systems Concepts and
Analysis Branch

Under authority of
H. L. Smith, Head
Surveillance Systems
Engineering Division

PK

# REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

1b RESTRICTIVE MARKINGS

2a SECURITY CLASSIFICATION AUTHORITY

2b DECLASSIFICATION DOWNGRADING SCHEDULE

3 DISTRIBUTION AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

4 PERFORMING ORGANIZATION REPORT NUMBER(S)
NOSC TR 995

5 MONITORING ORGANIZATION REPORT NUMBER(S)

6a NAME OF PERFORMING ORGANIZATION
Naval Ocean Systems Center

6b OFFICE SYMBOL *(if applicable)*
Code 721

7a NAME OF MONITORING ORGANIZATION

6c ADDRESS *(City State and ZIP Code)*
San Diego, CA 92152-5000

7b ADDRESS *(City State and ZIP Code)*

8a NAME OF FUNDING SPONSORING ORGANIZATION
Naval Material Command

8b OFFICE SYMBOL *(if applicable)*

9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c ADDRESS *(City, State and ZIP Code)*
Independent Research Programs
Washington, DC 20360

10 SOURCE OF FUNDING NUMBERS

| PROGRAM ELEMENT NO | PROJECT NO | TASK NO | WORK UNIT NO |
|---|---|---|---|
| 61152N | ZR0000101 | 632 | ZT10 |

11 TITLE *(Include Security Classification)*
IMPULSE: A CODE FOR HIGH-RESOLUTION IMPULSE RESPONSE FUNCTIONS IN RANGE-DEPENDENT DUCTS

12 PERSONAL AUTHOR(S)
R. C. Shockley, J. Northrop

13a TYPE OF REPORT
Final

13b TIME COVERED
FROM Oct 83 TO Aug 84

14 DATE OF REPORT *(Year, Month, Day)*
December 1984

15 PAGE COUNT

16 SUPPLEMENTARY NOTATION

17 COSATI CODES

| FIELD | GROUP | SUB-GROUP |
|---|---|---|
| | | |
| | | |

18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Acoustics,
SOFAR,
Range-dependent ducts,
Impulse response,
Ray-tracing

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

A computer code is reported which calculates the impulse response for long-range acoustic paths in SOFAR (Sound Fixing and Ranging) channels. Surface and bottom reflections are included. A single sound speed profile is treated and the bottom depth is allowed to be range-dependent. The impulse response is calculated from geometrical ray theory. It includes transmission loss, travel time, arrival angle, and bottom loss. The source may be located on the upper or lower boundary.

ORIGINATOR - SUPPLIED KEY WORDS INCLUDE:

20 DISTRIBUTION AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED UNLIMITED □ SAME AS RPT □ DTIC USERS

21 ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

Agency Accession
DN487 767

22a NAME OF RESPONSIBLE INDIVIDUAL
R. C. Shockley

22b TELEPHONE *(Include Area Code)*
(619) 225-7667

22c OFFICE SYMBOL
Code 721

# EXECUTIVE SUMMARY

## Objective

Develop high-resolution, ray-trace algorithms for the study of impulse response functi ons in underwater sound channels.

## RESULTS

1. A code has been written and documented illustrating the agreement with analytically tractable cases, representative plots of sound speed profiles and ray paths, and tabular output of arrival times, angles, and transmission loss.

2. Programs, collectively called IMPULSE, calculated impulse response functions for a source and receiver, one of which may be on a boundary of the duct, under the assumption of geometrical spreading loss.

3. These programs allowed transverse variation of the refractive index, and longitudinal variation of the width of the duct. Specular reflection is applied at boundaries, and bottom loss is included.

## RECOMMENDATIONS

1. IMPULSE should be generalized to include range-varying sound speed profiles and absorption loss.

2. Plotting options should be added to permit plotting of arrival angles and intensity versus time. Ray-path plots should be altered to achieve a set number of points regardless of step size.

3. Applications of the code in areas relevant to current Navy needs should be identified. IMPULSE should be applied to experimental programs for examining the predictability of actual ocean impulse response functions.

4. IMPULSE should be made known to research personnel at the Naval Oceanographic Research and Development Agency, Naval Postgraduate School, Fleet Numerical Weather and Oceanographic Center, and other facilities.

# CONTENTS

# ILLUSTRATIONS

ILLUSTRATIONS (Continued)

# I. INTRODUCTION

The FORTRAN code presented here calculates exact (within machine wordsize and storage limits) ray-theoretic impulse response functions. The impulse response consists of ray arrival angles, travel times, and geometrical ray-spreading transmission loss. Ray paths are calculated by numerical integration of a second-order ray equation of motion, and then used to determine flux tubes linking the source to the receiver, one of which may be on a boundary. The medium comprises 1) a flat, lossless, specularly-reflecting upper boundary or "surface," 2) a range-dependent, lossy, specularly-reflecting lower boundary or "bottom" 3) a depth-dependent, real index of refraction, and 4) an 1/R attenuation to account for cylindrical spreading.

A fourth-order Runge-Kutta method is used to integrate

$$d^2z/dx^2 = -[1 + (dz/dx)^2]c^{-1}\ dc/dz, \qquad (1)$$

where z (x) is the trajectory and c = c(z) is the phase velocity. Equation (1) may be viewed as the Euler-Lagrange equation implied by Fermat's principle, or simply a consequence of Snell's law. Piecewise continuous cubics are used to fit the index of refraction and the bottom profile, and then used to provide interpolated values between data points. The bottom reflection loss is linearly interpolated between given data points.

The user may enter arbitrary increments for 1) the range step size used in the numerical integration, and 2) the uniform angular separation between adjacent rays in the fan which originates at the source and is traced out to the range of the receiver. The limit angles in the fan are also user selectable. These features are useful in testing for the convergence of numerical results, and for detailed studies of the fine structure of arrivals at the receiver associated with a particular angular region of rays leaving the source.

This report presents the computational features of IMPULSE (ray tracing, surface and bottom reflections, etc.) and a few test cases of actual runs. It describes the use of the auxiliary plotting routine provided and the nature of pitfalls in applying IMPULSE to problems with undersampled environmental data or overly stringent ray density requirements.

# II. IMPULSE

## A. WHY A NEW RAY-TRACE CODE?

Frequently, analysis of propagation requires knowledge of the basic high-frequency, time-dependent, multipath arrival structure of signals for arbitrary source-receiver geometries in a particular analytically intractable waveguide. Such results are often good approximations to more exact wavelength-dependent modal solutions and are adequate to treat physical situations of interest, such as underwater sound propagation and tropospheric ducting of electromagnetic waves.

Many computer codes exist for these or similar problems. Assumptions made in such codes may, however, limit the generality of the media one can treat, or the location of the source or receiver. Codes which allow general media may involve a restrictive amount of computation or environmental data so that the results require considerable sophistication in their use and interpretation. Methods to reduce the number of calculations may reduce the accuracy or precision of the results because the impulse response may be evaluated only approximately.

To study the impulse-response-like arrivals of long-range underwater signals from shot-like sources (reference 1), we recently developed a FORTRAN code, called IMPULSE, which treats a depth-dependent sound speed and a range-dependent bottom profile. Although limited, as discussed later, ir the types of environments it can treat, IMPULSE does avoid most of the shortcomings mentioned above. Since IMPULSE may find diverse applications, this report has been prepared to give a complete description of its features and use. IMPULSE is not compared to other codes; instead, we establish the agreement between its calculations and analytic solutions for certain tractable cases. Also included is an example using a representative, "real-world" environment.

## B. GENERAL DESCRIPTION

IMPULSE computes almost exact solutions (limited by machine storage) for a large class of waveguides in the context of geometrical ray theory. It places minimal demands on users and provides results in a clear and simple fashion. For users with access to DISSPLA software, plots (of rays and the curve-fits to environmental data between points) are provided. For other installations, users may appreciate the option of "dumping" the information used for plotting for later access by plotting routines other than DISSPLA's. Compilation costs and basic storage needs are kept to a minimum by writing all ray-path information onto external files (if one wants to get plots) and using separately compiled plotting routines. Since only the point-to-point features of the propagation are relevant, no intermediate ray-path points need to be stored in arrays in core.

The basic environment treated by IMPULSE is as follows:

1. A flat, specularly reflecting upper boundary or "surface,"

2. A range-dependent lower boundary, or "bottom," described by up to 100 points

3. A depth-dependent (transversely-varying) index of refraction, described by up to 100 points, specified by the phase velocity of the wavefronts,

4. A source from which up to 400 rays may, with uniform but arbitrary angular separation, originate on either boundary,

5. A receiver which must not be on, or too near, a boundary (as discussed below), and

6. A bottom-loss function consisting of the loss in dB/bounce at 19 grazing angles.

IMPULSE traces rays from the source to the receiver's range using these data and additional input to govern the precision of calculations. It then evaluates the standard expression (reference 2) in ray theory for the transmission loss for each pair of rays (or each flux-tube) encompassing the receiver. This calculation, plus travel time and arrival angle, constitutes the impulse response. Phase information (and superposition of paths) is not carried out.

## C. RECIPROCITY AND RECEIVERS NEAR BOUNDARIES

It is important to note that the usual expression for transmission loss in geometrical ray theory assumes a cylindrically symmetric medium with the source on the axis of symmetry. This point bears directly on the question of reciprocity. In the simple case of a range-invariant medium, this symmetry is trivially satisfied, but if the bottom varies with range, this assumption of symmetry automatically precludes reciprocity. The problem of reverse transmission (receiver-to-source) involves a different environment, namely one in which the receiver is on the axis of symmetry. Therefore, no two-dimensional code with reciprocity corresponds exactly to a physical problem if range dependence is present. The relevance of reciprocity to the use of IMPULSE is simply the desire to analyze a receiver on the bottom and, therefore, invoke the principle of reciprocity: the multipath structure at the receiver from a suspended source is the same as that which would be seen if the suspended source were the receiver, and was used as a source. Within the tolerance of numerical errors, this type of reciprocity holds for IMPULSE simply because the ray tracing itself is carried out in a two-dimensional medium instead of in a three-dimensional medium. The cylindrical spreading factor 1/R is introduced purely artificially. In the two-dimensional medium, a reduced form of the full three-dimensional Helmholtz equation holds, that is, where $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial z^2$. The proof of reciprocity (reference 3) based on this equation, is still valid (with ray theory being the high-frequency limit).

The scattering of rays out of their initial plane of propagation still cannot be addressed using IMPULSE. Nor, for the reasons mentioned, and since IMPULSE uses the standard geometrical ray theory formula in analysis of range-dependent three-dimensional problems are the results strictly valid. Hence, the agreement between experiment and calculations will depend, in general, on the direction of propagation and bottom contour.

For many applications, it is reasonable to expect any discrepancy to be small. Analysis of the expected magnitudes is beyond our present aims. Similar comments naturally pertain to other ray or modal codes and it is clear that the inclusion of the realities of three-dimensional propagation constitutes an enormously more complex family of problems, although one which is indeed seeing progress (reference 4).

The reason for allowing only the source to be on the boundary in writing IMPULSE is worth noting — that there is no basic difficulty or special coding needed in tracing rays from the source. But, if the receiver is on the boundary, then the flux tubes encompassing the receiver necessarily hit this boundary. Therefore, the code would have to search for rays bracketing the receiver not in depth (which is easy), but along a generally irregular bottom (which is harder). It would also have to project the cross section of the flux tube normal to the eigenray, using rays at different depths and ranges, instead of just at different depths. For these reasons it was chosen to allow only the source to be on a boundary and to appeal to reciprocity for the cases where this direction of propagation is opposite to that in the physical problem under study.

The problem of a receiver too near a boundary is, of course, still present. Specifically, suppose we have a receiver near a boundary and trace a fan of rays at increments of 2°. Now it may turn out that no eigenrays exist, according to IMPULSE, because no rays were found to bracket the receiver's depth. It may be clear by inspection that eigenrays do exist. One solution is simply to determine roughly what rays at the source are near to eigenrays (from graphs or printouts) and perform higher resolution ray tracing of, say 0.2° increments in these angular regions. Since the rays are now closer together, it is more likely that a pair of them will encompass the receiver. If not, the procedure can be repeated indefinitely. Two pitfalls can occur: a) when the rays are so close that machine word size leads to numerical errors in calculating cross sections, and b) when the receiver is exactly on the boundary so that no rays can possibly bracket it in depth. It is not anticipated that these cases are likely to arise often. In such cases, it may be reasonable to simply model the problem as one where the receiver is, in fact, slightly away from the boundary.

## D. THE IMPULSE ALGORITHM

This section provides details on the computational methods in IMPULSE. For more complete information, the user should refer to the listing of IMPULSE in Appendix A. This listing may help the reader familiar with the general nature of IMPULSE in determining whether to carry out modifications in the code for more rays, refined bottom depth or bottom loss, surface reflection loss, volume attenuation, coherent summation, additional diagnostics, more calculations (e.g., phase integrals), or other specific needs (e.g., earth's curvature correction).

It may not be essential to read this section for successful application of IMPULSE. To make best use of the code, to appreciate problems that can arise in applications, and to design satisfactory input data sets, it may be helpful to know some computational details. This section also covers certain points relating to the accuracy of the code, introduces terms that might otherwise be unclear, and presents sign conventions, etc. The next section describes the various input variables and gives examples of runs.

It must be stressed that IMPULSE is basically a very simple code with highly restricted, modest aims. The only slightly subtle aspects of IMPULSE, perhaps, are its algorithms for reflections, and its need for adequately sampled data for the index of refraction and bottom depth. The code uses cubic polynomials for phase velocity and bottom profiles. The piecewise continuous curves fit four data at a time for interpolation between the data. The use of cubics leads to continuously varying functions with curvature, rather than linearly varying functions. (The use of cubics allows a sparser sampling of data than would linear segments to obtain equivalent accuracy in approximation of the original functions. Cubics can also lead to wild overshooting between data if the original curve is undersampled.) There are several miscellaneous details concerning the control of precision of calculations, plotting, and diagnostics. Some users will find the use of dimensionless variables confusing at first. Users with electromagnetic applications may find the acoustics context disconcerting, as we have used the term "sound speed" for phase velocity in the remainder of the report as a remnant of our initial area of study. Additionally, users with applications other than acoustic may need to (re)normalize their data in a new system of units simply to conform to IMPULSE's printing format specification, or, alternatively, modify the width of the print fields in IMPULSE to accommodate the conventional system of units.

## 1. General Comments

IMPULSE is written in FORTRAN V. All variables are double precision. It consists of two executable main routines and seven subroutines. The latter are two routines for curve-fitting the sound speed profile (SSP) (or phase velocity) and bottom depth, a function evaluation (the right hand side of Equation (1)), a cubic fit to the four points on the ray trajectory if a bottom reflection occurs, a bottom reflection loss interpolation, and two routines for dumping the SSP and bottom depth curves (200 points are calculated and dumped) onto a file for plotting. The plot routine is a separately executable program from the main routine The total core storage required is approximately 14K words for the main routine, and 50K for the plotting routine, after compilation. No external files are required if printed output alone is desired. For plots, an external file is needed with unit number 9 attached to it for dumping data. After the run is completed this file must be renumbered as external unit 10 for the plotting routine.

## 2. Dimensionless Variables

Consistent length units must be used throughout. If the SSP is in m/s, depths must always be in meters. It is not permissible to mix feet and meters, or meters and yards, etc., in one run.

## 3. Input

The input variables are described more completely in the next section. The data consists of:

1. The SSP as (depth, sound speed) pairs.

2. A switch for getting the SSP plot.

3. The bottom depth data as (depth, range) pairs.

4. A switch for getting the bottom depth plot.

5. The bottom loss in dB/bounce at 19 points in (bottom loss, grazing angle) pairs.

6. A range step size.

7. The source depth.

8. The receiver depth.

9. The range to the receiver.

10. The angular spread of and separation between rays leaving the source.

11. A tolerance in the precision of locating bottom reflection points in range.

12. The number of the ray whose trajectory is to be printed, along with other diagnostic data, if desired.

13. A switch for getting plots of rays, and

14. The numbers of the rays to be plotted if plotting is desired.

## 4. Ray-Tracing Method

Since IMPULSE is basically short and simple, we present its theory in essentially complete form.

Snell's law is usually expressed as (figure 1)

$$n_1 \sin \phi_1 = n_2 \sin \phi_2. \tag{2}$$

Here, $n_i$ and $\phi_i$ are the index of refraction and the angle between the ray and the normal to the surface, respectively, in medium i.

If the complement of $\phi_i$ is called $\theta_i$ and $n_i$ defined as $c_o/c_i$ (where $c_o$ is a reference sound speed and $c_i$ the local sound speed in medium i), Equation (2) becomes

$$c_1/\cos\theta_1 = c_2/\cos\theta_2 \equiv c_m \qquad (3)$$

A coordinate system with $dz/dx = \tan\theta$ is now introduced and the layered medium treated as a continuum; that is, $c = c(z)$. We define the constant $c(z)/\cos\theta(z)$ as $c_m$ (m stands for maximum), the limit-speed the ray can reach, at which depth it undergoes total internal reflection. Note that z increases in the downwards direction; $\theta$ is negative for an up-going ray.

Squaring Equation (3) and rearranging, one gets

$$(dz/dx)^2 = c^2_m/c^2(z) - 1. \qquad (4)$$

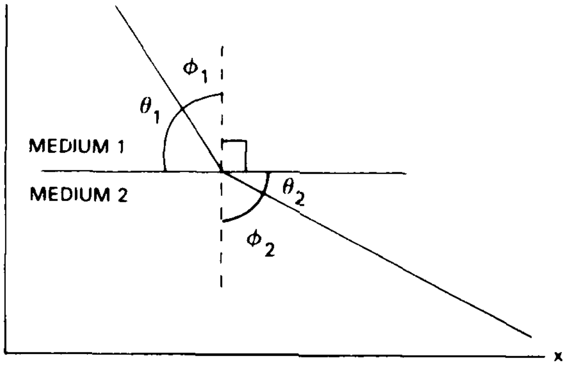


Figure 1. The coordinate system.

Differentiate Equation (4), obtaining

$$2(d^2z/dx^2)(dz/dx) = -2[c^2_m/c^3(z)](dc/dz)(dz/dx)$$

or

$$d^2z/dx^2 = -[c^2_m/c^2(z)][1/c(z)]dc/dz.$$

Eliminate the first factor on the right-hand side via Equation (4) to find

$$d^2z/dx^2 = -[1 + (dz/dx)^2]\, c^{-1}\, dc/dz, \qquad (5)$$

which can be solved via Runge-Kutta methods, given c(z).

Runge-Kutta numerical integration consists, briefly, of computing solutions to equations such as

$$d^2y/dx^2 = f(y,\, dy/dx), \qquad (6)$$

given $y(x_o)$, at points $x_o + h$, $x_o + 2h$, ... , by evaluating f using weighted sums for the arguments based on the current values of x, y, and dy/dx. By a judicious choice of the weights, one can obtain accuracy to a high order of h.

Reducing h shows whether or not the numerical solution has converged on the analytic solution.

Specifically, Equation (6) can be solved numerically as follows (reference 5): Let $y' = dy/dx$, and let $y_n$ be the numerical solution at $x_n = x_o + nh$. Let the numerical solution at $x_{n+1}$ be given by

$$y_{n+1} = y_n + h\, [y'_n + (k_1 + k_2 + k_3)/6], \tag{7a}$$

$$y'_{n+1} = y'_n + (k_1 + 2k_2 + 2k_3 + k_4)/6, \tag{7b}$$

where

$$k_1 = hf\, (y_n, y'_n), \tag{8a}$$

$$k_2 = hf\, (y_n + 1/2\, hy'_n + hk_1/8,\ y'_n + k_1/2), \tag{8b}$$

$$k_3 = hf\, (y_n + 1/2\, hy'_n + hk_1/8,\ y'_n + k_2/2), \tag{8c}$$

$$k_4 = hf\, (y_n + hy'_n + hk_3/2,\ y'_n + k_3). \tag{8d}$$

The difference between $y_{n+1}$ and $y(x_{n+1})$ is proportional to $h^5$. In other words, the Taylor series expansion of $y_{n+1}$ agrees, up to and including terms of order $h^4$, with the Taylor series for $y(x_{n+1})$. Proof of this statement is algebraically straightforward, but lengthy, and is, therefore, omitted.

The motivations for using the above approach are its easy implementation and the lack of singularities, or numerically ill-conditioned problems, at the points of total internal reflection. Using linearly segmented SSPs, where rays are arcs of circles, is also feasible but always biases the results; the SSP linear segments lie inside every region of curvature. Linearly segmented SSPs also lead to more complex coding in evaluating the trajectory segments.

For each ray in the user-defined fan leaving the source, IMPULSE applies the above method to calculate the depth and angle of the ray at the range of the receiver. The range step size, h, is selected by the user. Reflections from boundaries require special treatment.

It is stressed that each ray is traced completely from the source to the range of the receiver before the next ray is traced. The endpoint of each ray is stored as well as the slope at the endpoint and overall travel time. These stored quantities are needed to compute the transmission loss and arrival times.

Unlike sophisticated differential equation-solving codes, IMPULSE does not automatically estimate the error in its numerical solutions or subsequently adjust the step size to meet user-defined criteria. IMPULSE needs the user to define a step size and uses that value throughout the computation.

## 5. Reflections at Boundaries

IMPULSE uses different methods to reflect rays at the surface (z = o) and bottom. The method for surface reflection actually affects the input data; it requires that the image of the first three data points on sound speed below the surface be provided as input. That is, the first, second, and third data must equal the seventh, sixth, and fifth point except with negative depths. The fourth datum is the sound speed at the surface.

After a ray's depth is updated (advanced in range) the code asks if the new depth is negative. If it is, the ray has passed through the surface in that range step. Since the SSP is symmetric about the surface, all that is required then is to reverse the sign of the ray's depth and slope at the updated point to reflect the ray from the surface. In other words, the image of the physically reflected ray is actually traced first and then brought back to physical space.

After a ray's depth is updated, and if it has not crossed the surface, the code asks if the new depth exceeds the depth of the bottom at the new range. If it does, a special segment of code is called into play, and the intersection of the local ray path with the bottom is iteratively determined. The code does not magically stop ray tracing at the bottom (nor at the surface) but completes the update using the SSP just as if the boundaries were absent.

The iterative procedure works by retaining the ray's depth at four points — namely, the current and updated values, plus the two previous depths. When the code finds that the ray has passed through the bottom, it fits a cubic to these four points and then determines the point of intersection of this cubic and the bottom. The iteration consists of working forward from the current range value (before the ray depth exceeds the bottom) in steps one-tenth of the range step, until the depth of the cubic exceeds the bottom. Then it cuts the iteration step size in half, reversing its sign, and works backwards until the ray path is above the bottom, and so on. The procedure converges to a point of intersection, since both cubics are continuous, but stops when the iteration step size becomes less than a user-specified value.

After the point of intersection is determined to the desired accuracy, the ray path's slope and the bottom slope are found. Specular reflection is applied and bottom loss is calculated and added to the previous bottom loss in decibel units.

Finally, the difference between the end of the range step and the point of intersection is calculated (that is, the distance the ray must go to complete the present range step). A second Runge-Kutta integration is applied over this distance and the code resumes normal update procedures. For example, if the bottom were hit one-half of the way through the range step, the second Runge-Kutta integration would update the ray over one-half a range step from the intersection point to the end of the range step.

6. Travel Time

Once the ray's depth is updated, IMPULSE updates the accumulated time along its path since leaving the source. The arc length is just

$$[(dz)^2 + (dx)^2]^{1/2}$$

where dz and dx are the change in depth and the range step size respectively. The speed locally is taken to be the average of the sound speed at the current depth and updated depth.

7. Transmission Loss

IMPULSE traces each ray within a fan of rays leaving the source (at user-selected increments in angle and between user-selected limit angles) out to the range of the receiver, one ray at a time.

As IMPULSE traces a given ray, it saves (in array elements) only the current depth and slope, plus the previous two depths (as variables to be used, if needed, for calculating bottom reflection points), and accumulated travel time. Once a given ray is completed, IMPULSE begins tracing the next ray and continues until all rays are traced to the receiver's range.

At this point, the depth and slope of each ray at the receiver range are stored in an array, as well as the "launch" angle of each ray at the source. These numbers suffice to evaluate approximately the usual expression (reference 2) for ray-spreading transmission loss at the receiver, provided at least one pair of rays, which were adjacent at the source, bracket the receiver's depth. This requirement is the weakest constraint for approximating the derivative of depth with respect to launch angle by a ratio of differences in depths and launch angles. If a pair of rays not adjacent at the source bracket the receiver, there is no assurance that the depth is a monotonic function of the launch angle over the region of values at hand. Therefore, there is no reason to say the ratio of differences is approximately the derivative.

Transmission loss is defined as the (decibel) ratio of the intensity at the receiver, and at a unit distance from the source. The formula for transmission loss (reference 2) is

$$TL = -10 \log [R^{-1}(\cos \theta_s/\cos \theta_R)(dz/d\theta_s)^{-1}{}_R], \qquad (9)$$

where R is the receiver's range (measured in units of the reference distance from the source at which the reference intensity is defined), $\theta_s$ is the angle of the ray at the source, $\theta_R$ is the angle at the receiver, and $(dz/d\theta_s)_R$ is the derivative of the ray's depth with respect to its launch angle at the receiver. Equation (9) is easily derived by considering two rays leaving the source at angles $\theta_s$ and $\theta_s + d\theta_s$, respectively. Their depth difference at the receiver is $(dz/d\theta_s)_R \, d\theta_s$. Projecting this depth difference onto the cross section of the flux tube introduces a factor cos $\theta_R$. The factor cos $\theta_s$ arises from the expression for the solid angle subtended at the source by the rays. Conservation of energy leads to Equation (9); that is, the product of the intensity and the flux tube's cross sectional area is a constant.

In approximating Equation (9) numerically, both the receiver and source angles are taken to be the average of the angles of the two rays. The derivative is just the absolute value of the ratio of depth difference and the increment in source angle.

After Equation (9) is approximately evaluated, the accumulated decibel losses from bottom reflection for each ray are added. Since the two rays defining the flux tube do not generally have identical losses, their losses are power-averaged.

## 8. The Use of Cubics

IMPULSE uses piecewise continuous cubics to fit the SSP and bottom depth data. The derivative in both functions is, therefore, discontinuous at the meeting point of two cubic segments. This is not an inherent shortcoming or weakness in the code. The Runge-Kutta formula always has a well-defined gradient to work with, as does the bottom reflection algorithm. If spline fits were used, an artificial curvature would arise in fitting a polynomial to regions of linear variation following regions with curvature, simply because the derivative would be forced to be continuous. In our case, the cubic segments lead to discontinuous derivatives. The value of the discontinuity can be easily made very small simply by adequately sampling the curve being fit.

The cubic approximations can introduce large errors in cases where the cubic is forced to match a very sharp bend. Although the curve always passes through the given data, it may differ greatly from what one's eye determines to be a reasonable curve between data points. For example: if three SSP data are widely spaced and colinear, and the fourth point is only slightly below the third but with a much different value, then the resulting cubic through these four points will oscillate wildly whereas it should be constant near the first three points.

The presence of such unreasonable fluctuations is readily detected and eliminated by means of plotting routines for the curve fit representations of the SSP and bottom depth functions, and the subsequent introduction of additional data points, if needed, which lead the curve smoothly around the bend. The added points effectively force the cubic fits closer to the underlying curves. In practice, one can ensure reasonable-looking curves by making certain that any regions of large curvature are specified by at least four data points.

The cubic fit routines effectively break the data into layers with four points in each layer and one point in common where layers meet. The first layer consists of data points (1,2,3,4) and the second consists of data points (4,5,6,7) etc. Beyond the last compete layer, linear segments are used to interpolate between data. Beyond the last data, the slope is defined to be zero. Therefore, the index of refraction must be specified by input data to, at least, the depth of the bottom at its deepest to ensure that any unrealistic extrapolation is avoided. Similarly, the bottom depth must be specified by input data extending to at least the range of the receiver to avoid unrealistic extrapolations.

## 9. Plotting Options

IMPULSE has an independently executable plotting routine that uses the DISSPLA software. To obtain plots, a file must be available for IMPULSE to write onto as external unit number 9, and certain so-called input switch variables (because the only values recognized are 0 and 1) must be set to 1. Following the IMPULSE run, the code PLOTS is run. It will read all needed data from the previously used file as external unit number 10 and requires no other data. In other words, the input to IMPULSE also defines each run of PLOT.

Plots are indispensible aids in debugging runs as well as in interpreting the results. Plots show whether the cubic polynomial segments are good representations of the intended curves, where eigenrays lie, and where bottom reflections occur, etc.

IMPULSE allows users to plot the piecewise continuous cubic approximation of the SSP at 300 points uniformly spaced between two arbitrary, user-specified depths. This is obtained by setting an input variable called IPLT1 to unity and entering two depths between them. 300 SSP points are calculated by IMPULSE and dumped together with the plot control switches.

IMPULSE allows users to plot a 300-point curve of the cubic polynomial's fit to the bottom depth as a function of range. In this case, the points automatically extend from the source range (zero) out to the receiver's range. The switch variable is IPLT2.

Finally, IMPULSE dumps ray trajectory data, if desired, as the rays are traced. If PLT3 is entered as unity, then the user must specify how many rays are to be plotted and what number position they are in the fan (e.g., 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 30, 31, 32, 33, 34, 35 or 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, etc.). If IPLT3 is entered as unity, IMPULSE automatically sets IPLT2 to unity also so that the trajectories are always shown in relation to the bottom.

## 10. Diagnostics

IMPULSE has a provision for printing the evolution of up to 100 ray paths in the fan. The output also shows the ray's slope at each point in range, the bottom depth, and bottom slope. At bottom reflections, the ray slope, bottom slope, and reflected slope are also dumped, plus the iteration history.

In addition to the user-directed, ray-tracing diagnostic details, IMPULSE has a small set of standard diagnostic tests with associated messages designed to detect improperly entered data sets, and pathological conditions arising during ray tracing, etc. These are:

a) A test on the number of SSP and bottom profile data which must lie between 4 and 100 inclusive;

b) A test on bottom-reflected rays which detects intersection angles less than zero (the ray is approaching the bottom from below) or greater than 90° (the ray is normal to the bottom and would be reflected backwards);

c) A test on the number of eigenrays which prints a message if no flux tubes bracket the receiver; and

d) A test on the bottom loss evaluation which prints a message if the grazing angle (complement of the angle of incidence) is less than the first angle at which bottom loss is given in the input data or is greater than the maximum angle.

## 11. Recommended Steps in Using IMPULSE

No guaranteed algorithm exists to handle every possible set of environmental data. In certain cases no eigenrays may be present. Other cases may require very careful searches for eigenrays, or may involve great sensitivity to small changes in the environmental data or the source-receiver geometry. In the majority of cases, however, one can expect to obtain reasonable results with modest effort. It is still worthwhile, nevertheless, to list certain basic steps a user typically must take in calculating eigenrays and their associated transmission loss. (For a specific case study in which these steps are followed, the reader is referred to Section IIIC.) The general steps are as follows:

a) Examination of environmental data. Prepare a data set consisting of all environmental data but with very few rays requested. Let IPLT1 = 1 and IPLT2 = 1. This creates plots of the SSP and the bottom profile. Execute the adjunct plotting program to produce the plots. Examine the plots to see if the piecewise continuous cubic fits to both functions are satisfactory. If not, add data points in the regions where the cubic disagrees with the intuitively obvious underlying curve, forcing the fit into better agreement, and examine the new curves. When the agreement is satisfactory, eigenrays may be sought.

b) Search for eigenrays. Turn off the switch variables for plotting the SSP and the bottom, and set IPLT3 = 1 to get plots of the bottom and of the ray paths. Select a wide range of initial ray angles, a large increment between rays at the source, and a large step size. From the resulting ray paths' plots, or from the printout of the rays' depths at the receiver range, determine the initial angles where eigenrays are likely to lie. The angular increment and step size should be large enough that the runs made at this stage are not prohibitively expensive and small enough to give approximately correct results. Once a family of potential eigenrays are found, they may be computed to greater precision.

If the search does not suggest any likely candidates, one may slightly move the receiver or source to attempt a more favorable geometry. Typically, this means moving the receiver away from a boundary (see Section IIB). If it proves difficult to find candidates after a detailed search, this suggests that no eigenrays exist. Physically, the signal reaches the receiver via an evanescent wave or diffraction, or via other than great circle paths. In this case, geometrical ray theory cannot provide answers and one must resort to codes which include wavelength dependence.

c) Detailed eigenray computation. For the angular regions where eigenrays are expected, run IMPULSE with a fine angular increment and a fine range step size. For each pair of rays which are adjacent at the source and bracket the receiver, examine the transmission loss and travel time for decreasing angular increment, range step size and, perhaps, DXBOT. When the results cease to vary when these parameters are further reduced, convergence is obtained. Studies of convergence are in Sections IIIA and IIIB.

If results are unstable, examine the ray paths and environmental data for the presence of any possible sources of this behavior. These may involve, for example, the presence of a knee in the SSP at the exact depth of the receiver, or a caustic surface near the receiver. Another potential source of error can be too fine a step size; while it has not been seen in any studies to date, there is the possibility that accumulation of roundoff error over a great many steps can reduce the accuracy of IMPULSE. As a crude rule, we suggest that approximately 100 steps be allowed per loop length.

## E. INPUT

All input is read under free-field format. Here, we assume that cards are used, but in the case studies, data are from an external file (unit 8) as in the listings in Appendix A. The reader may refer to a FORTRAN programmer's reference manual for the system to be used to obtain complete details on free-field format. However, a short description is provided here. The free-field format is essentially the simplest specification. If several data are on one card, they only need to be separated by commas. They do not have to occupy special fixed fields (i.e., columns) on the card. They can be punched in engineering or scientific notation. Although the variables read in are double precision they do not have to be represented as double precision on the cards; free-field input performs mode conversion. If the data read by one read statement extends over many cards, as usually happens for SSP data and bathymetry, the end of a card is a legal delimiter between data items replacing the comma. Therefore, you must not put a comma after the last item on a card. This would indicate to the program that another item is to follow on that card and would result in an execution error. If input from another device is desired, the user must change the unit number from 5 to the appropriate value in the read staements in the main routine, PULSE. This was done for some of our test cases.

Before describing the input data, we first define several FORTRAN variables which are read in from cards.

a) NPTS1 is the number of points in the input sound speed (or, more generally, phase velocity) profile including the three values above the surface which create a mirror image of those below the surface (see section IID.5 on reflections at boundaries).

b) IPLT1 is a switch or control variable which, if set equal to unity, causes a set of 300 points on the SSP to be computed evenly spaced between two user-selected depths, after the curve-fit routine has been called, and dumped along with their depths onto external unit 9. If not equal to one, IPLT1 has no effect.

c) ZZ(I) is an array of NPTS1 depths where the input SSP data are given. The first three (if surface reflections are of interest), must be the negatives of the seventh, sixth, and fifth, respectively (see Section IID.5). I must be between 4 and 200, inclusive.

d) C(I) is an array of sound speed values corresponding to depths ZZ(I). (See remark at the end of Section II.D.8 regarding the extrapolation algorithm.)

e) Z1 and Z2 are the depths between which the SSP is computed from the cubic-fit routines if a plot of the SSP is desired (that is, if IPLT1 = 1). An array of 300 values from Z1 to Z2 in depth is dumped onto external unit 9 if IPLT = 1.

f) NPTS2 is the number of bottom profile points in the input data.

g) IPLT2 is a switch variable which, if set equal to unity, causes a set of 300 depths on the bottom to be computed. This is done after the bottom curve fitting routine is called, between zero range and the receiver range, and dumped along with their ranges onto external unit 9. If not equal to one, IPLT2 has no effect.

h) RB(I) is an array of ranges at which the bottom depths are input.

i) ZB(I) is the array of bottom depths corresponding to ranges RB(I).

j) BOTLOS(I) is an array of 19 bottom loss values in dB/bounce.

k) BOTANG(I) is the array of 19 grazing angles (in degrees) corresponding to BOTLOS, for example 0, 5, ..., 85, 90.

l) DX is the range increment for the Runge-Kutta routine. DX should divide R (see below) an integer number of times with a small positive remainder.

m) ZS is the source depth.

n) ZR is the receiver depth.

o) R is the horizontal distance from source to receiver.

p) THETA1 is the angle (in degrees) of the first ray leaving the source, measured with respect to horizontal.

q) DTHETA is the increment (in degrees) in the angles at which rays leave the source.

r) THETA2 is the angle of the last ray to leave the source.

s) DXBOT is a cutoff value for iterating on the point of intersection of a local ray segment and the bottom.

t) ITRACE is the number of a ray whose trajectory is to be printed for diagnostic purposes. If ITRACE = 1, the ray at THETA1 is printed, for example. If ITRACE = 0, no trajectories are printed.

u) IPLT3 is a switch variable. If set equal to one, it causes a set of ray trajectories to be output to unit 9 for subsequent plotting. Otherwise it has no effect.

v) NPATHS is the number of ray paths to be output to unit 9 for subsequent plotting.

w) IPATH(I) is an array of the numbers of rays whose trajectories are to be output to unit 9 for subsequent plotting.

Note in the above description that certain variables are irrelevant if certain of the switch variables are not unity. If IPLT1 = 0, then Z1 and Z2 are not needed. If IPLT3 = 0, then NPATHS and IPATH are not needed.

The order in which these input data must appear on cards is now described. (IMPULSE can be modified easily to read from any other external unit than the card reader simply by changing the external unit number from 5 to the appropriate unit number.) By card number, the input is:

1. NPTS1, IPLT1

2a. ZZ(1), C(1), ..., ZZ(NPTS1), C(NPTS1)

2b. If IPLT1 = 1, Z1, Z2

3. NPTS2, IPLT2

4. RB(1), ZB(1), ..., RB(NPTS2), ZB(NPTS2)

5. BOTLOS(1), BOTANG(1), ..., BOTLOS(19), BOTANG(19)

6. DX, ZS, ZR, R, THETA1, DTHETA, THETA2, DXBOT

7a. ITRACE, IPLT3

7b. If IPLT3 = 1, NPATHS

7c. If IPLT3 = 1, IPATH(1), ..., IPATH(NPATHS)

# III. CASE STUDIES

It is essential to examine the accuracy of IMPULSE as a function of the various input variables to demonstrate that it does indeed converge to analytical answers with sufficiently detailed input data. It is also essential to establish the typical sizes of the "error" for typical values of the number of SSP data, range step size, and ray angular spacing. A detailed numerical analysis is not presented here since we are not as interested in details of the various machine-dependent tradeoffs in truncation and roundoff errors, as in the basic behavior of the code in comparison to analytic solutions.

The following two sections treat, respectively, a Hirsch SSP (reference 6) whose rays are sinusoids, and a parabolic bottom underlying a constant-speed medium. By suitable source and receiver positioning, it is possible to find simple analytic expressions for transmission loss and signal speed (range divided by travel time) for the eigenrays. These quantities are the essential descriptors of the impulse response function for that geometry. Their analytic values are compared to the numerical results for separate variations of NPTS1, DX, and DTHETA which are, respectively, the number of SSP data, the range step-size, and the angle between successive rays leaving the source.

It is also essential to study at least one example of a typical real-world SSP, including interaction with boundaries. The third section to follow presents a study of the response at a bottom-mounted hydrophone to a near-surface source for a representative SSP from the Colborn-Wright data base of North Pacific Ocean profiles (reference 7), using a hypothetical bottom contour and bottom loss function.

## A. CASE STUDY OF TOTALLY REFRACTED RAYS USING THE HIRSCH PROFILE

The SSP (reference 6)

$$c(z) = c_o/[1 - (z-z_o)^2/a^2]^{1/2} \tag{10}$$

leads to ray paths which are sinusoidal. This is demonstrated by writing Equation (5) as

$$\begin{aligned} d^2z/dx^2 &= -(1 + \tan^2\theta)\, c^{-1}\, dc/dz \\ &= (d/dz)(c^2_v/2c^2) \end{aligned} \tag{11}$$

where $c_v = c/\cos\theta$ is the sound speed at the "vertex," or point of greatest off-axis excursion. From Equations (10) and (11),

$$d^2z/dx^2 = -\,(c_v/c_o)^2(z-z_o)/a^2, \tag{12}$$

which shows that the ray obeys the same equation of motion as a simple harmonic oscillator.

For simplicity, we place both the source and receiver on the sound channel axis and select the range to equal one cycle distance of the trajectory.

For the initial condition where the ray is at the axis, $z(o) = z_o$, the solution of Equation (12) is

$$z-z_o = A \sin [(c_v/c_o)x/a]$$

$$= A \sin (x/a \cos \theta_o), \tag{13}$$

where A is a constant to be determined and $\theta_o$ is the angle of the ray at x = 0. Hence, the period of the trajectory (or loop length) is

$$x_p = 2\pi a \cos \theta_o, \tag{14}$$

which decreases monotonically with increasing $\theta_o$. At x = 0, the slope is tan $\theta_o$. Differentiating Equation (13) and setting x = 0 therefore yields

$$\tan \theta_o = A/a \cos \theta_o$$

or

$$A = a \sin \theta_o \tag{15}$$

which shows that the amplitude of the path (the maximum off-axis excursion) increases as the sine of the initial angle $\theta_o$. The final form of the solution is, from Equation (13) and Equation (15),

$$z = z_o + a \sin \theta_o \sin (x/a \cos \theta_o) . \tag{16}$$

To evaluate the transmission loss in Equation (9), it is necessary to have an expression for the derivative of the depth of a ray at a fixed range with respect to the launch angle at the source. From Equation (16) this is, at a range of one loop length ($x_p$),

$$(dz/d\theta_o)_{xp} = 2\pi a \sin^2 \theta_o /\cos \theta_o,$$

leading upon substitution into Equation (9) to

$$TL = 20 \log (2\pi a \sin \theta_o). \tag{17}$$

Travel time or signal speed is also of interest. By definition, signal speed is the range divided by the travel time of a ray. Signal speed is a function of source and receiver positions. For the case when both the source and receiver are on the sound channel axis, it can be shown that the signal speed is

$$\bar{c} = 2 c_o \cos \theta_o /(1 + \cos^2 \theta_o). \tag{18}$$

For the numerical case study, we choose $z_s = z_o = 2000$, $a = 3000$, $c_o = 1480$. The channel width parameter, a, and the axial speed, $c_o$, are roughly representative of actual deep ocean values in meters and meters/s, respectively; the SSP is quite unrealistic for actual cases. The way IMPULSE converges to the analytic results for this case should give considerable insight into its behavior in more general (analytically intractable) cases.

The range is selected to be one cycle distance of a ray leaving the source at 15° depression angle, so that $R = x_p = 18207.3$ (Equation (14)). To facilitate the execution of test case runs, a special routine called TEST was written (see Appendix B) which builds a Hirsch SSP data file for IMPULSE but in itself needs only general instructions such as the number of SSP data, $z_o$, a, R, DX, and plotting switches.

Figures 2 and 3 show the SSP and a set of ray paths for this case. In figures 4 through 8, we present a comparison of the numerical results and the analytic results. The analytic result for transmission loss (Equation (17)), is TL = 73.8 dB and for signal speed (Equation (18)), it is $\bar{c} = 1479.1$.

Figure 4 shows that, except for residual errors from other parameters, the code converges to the analytic value for TL if 30 points are input to describe the SSP. The anomaly at NPTS1 = 14 is not understood, but, presumably, it arises because a layer in the piecewise cubic fit to the SSP ends at the channel axis and another begins. As noted in the figure caption, DX = 303.45 and DTHETA = 0.1°. With this value of DX, IMPULSE makes 60 range steps to reach a range of 18207, which is 0.3 away from the range where analytic results are evaluated. Therefore, we have 60 range steps per loop length.

Figure 5 shows that the numerical results for TL converge to the analytic results when as few as 8 range steps are taken per loop length. In this case, NPTS1 = 100 and DTHETA = 0.1°.

Figure 6 presents a study of the sensitivity of signal speed results to DX. The same parameters are used as for figure 5. Convergence is obtained for DX less than approximately 0.025 $x_p$ (i.e., 40 range steps per cycle).

Figures 7 and 8 show the effect of varying the spacing of rays at the source, DTHETA. The plots show a pair of sudden jumps in the numerical results. Examination of the output from IMPULSE (not shown) in the neighborhood of these jumps explains the behavior. Since IMPULSE searches for rays which bracket the receiver, it is very sensitive to slight changes in inputs which can switch the particular rays bracketing the receiver. At the position of the jumps, the detailed output from IMPULSE shows that the pair of rays found by IMPULSE does indeed hop, so that the TL or signal speed is basically calculated on rays largely above the receiver and then largely below the receiver. In other words, the receiver lies on the extreme edge of a ray pair. The precise position of such jumps depends on how one defines the fan of rays (the limit angles θ1 and θ2) and their spacing. Convergence to analytic results (to within residual errors) is obtained for angular spacing less than or approximately equal to 0.2°.

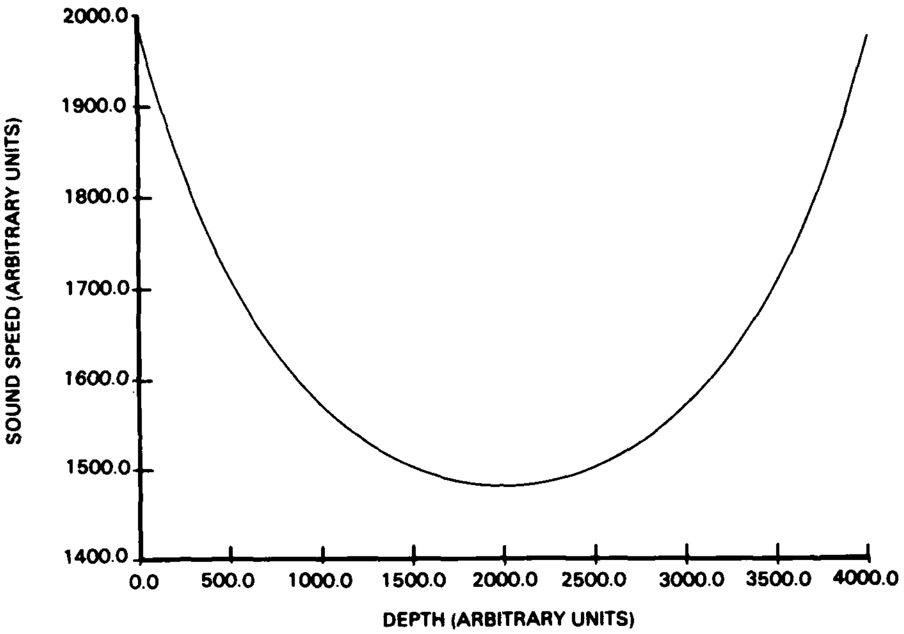


Figure 2. The SSP of equation 10 is shown for the case $z_0 = 2000$, $a = 3000$.

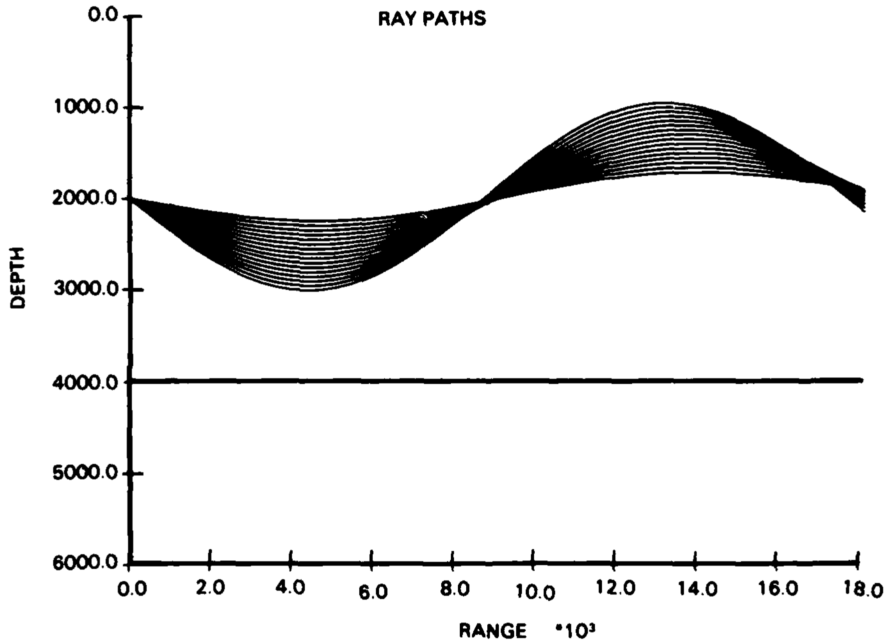


Figure 3. Ray paths are shown for $\theta 1 = 5°$, $\theta 2 = 20°$, $D\theta = 1.0°$, DX = 364.14, and NPTS1 = 100.

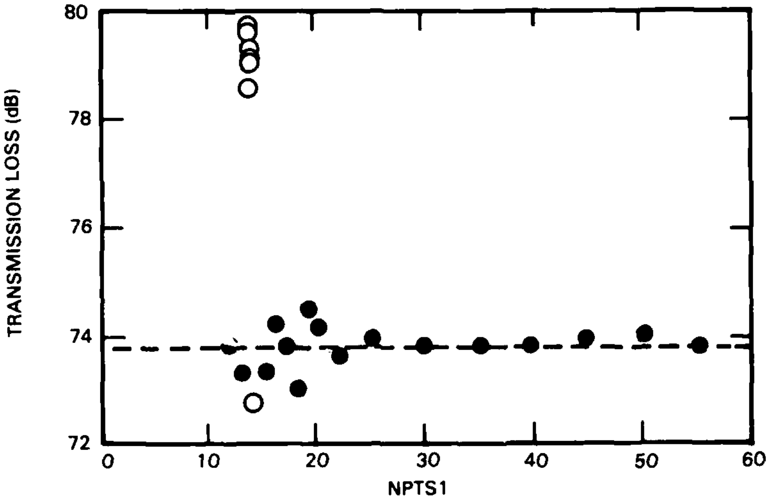


Figure 4. The numerical values (dots) are compared to the analytic value (dashed line) of transmission loss for various values of the number of SSP data. The anomaly at NPTS1 = 14 is not understood. The step size was DX = 303.45 (R/DX ~ 60) DTHETA = 0.1°. Circles represent variations of DX for fixed NPTS1 = 14. Residual errors probably arise from DX and DTHETA.

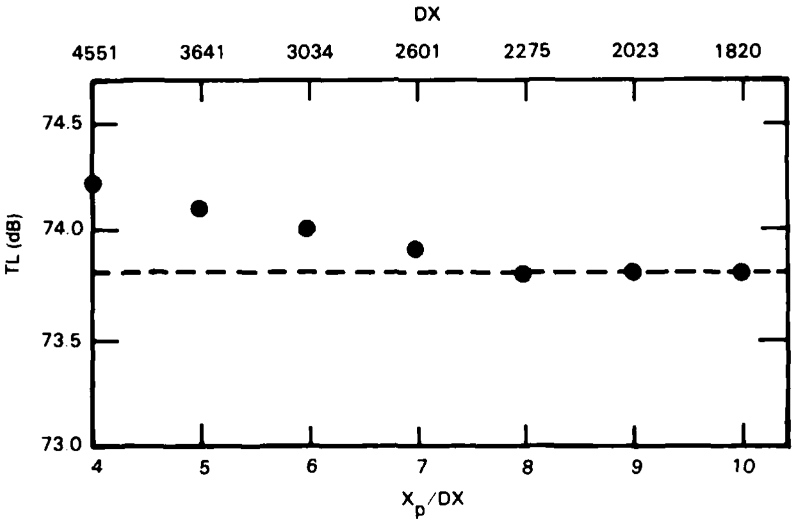


Figure 5. The numerical transmission loss (dots) is compared to the analytic result (dashed line) for several values of DX, the range step size, corresponding to 4, 5, ..., 10 steps in a cycle of the trajectory. For these results, NPTS1 = 100 $D\theta = 0.1°$.

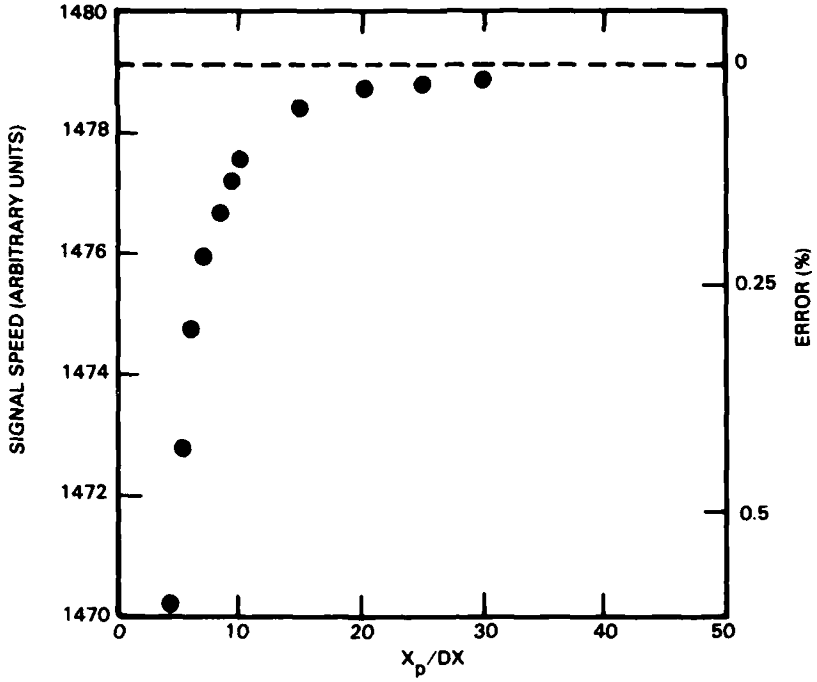


Figure 6. The numerical values of signal speed (range divided by travel time) are compared to the analytic result (dashed line) for varying range step size in the Runge-Kutta solution for the ray path. For these runs NPTS1 = 100 and DTHETA = 0.1°.

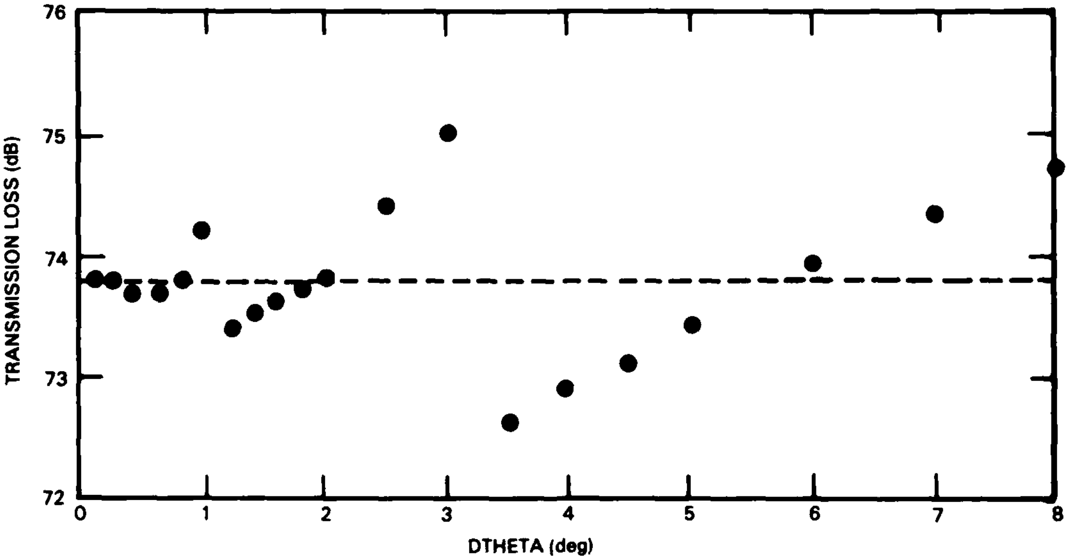


Figure 7. The numerical values of transmission loss (dots) are compared to the analytic result (dashed line) for variations in DTHETA. (The jumps at 1° and 3° result from the combination of DTHETA and the angular range limits THETA1 and THETA2, which leads to switches from eigenray pairs largely above to largely below the receiver.)

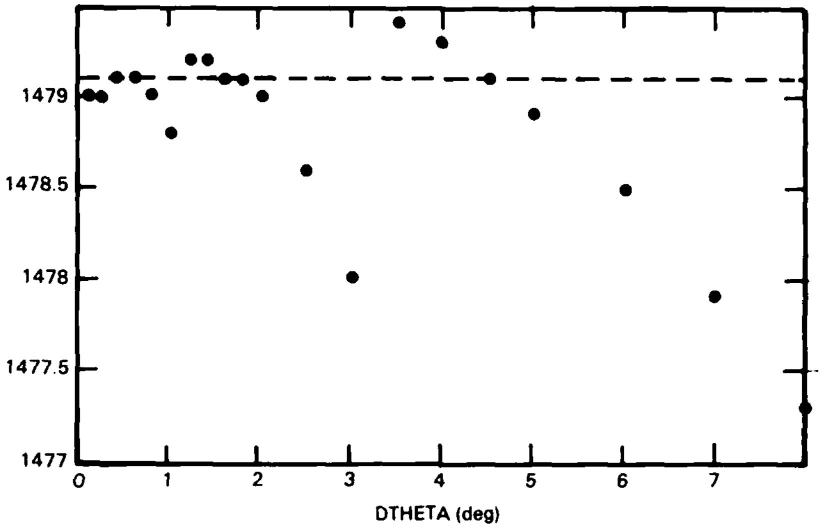


Figure 8. Numerical results for signal speed (dots) are compared to the analytic result (dashed line) for various values of DTHETA, the angular increment between adjacent rays leaving the source. For these results, NPTS1 = 100 and DX = 364.14. (Jumps at 1° and 3° arise from the choice of DTHETA and the limiting angles of the fan of rays at the source.)

It is worthwhile to point out that these results are specific to the particular SSP studied. For other cases one should generally anticipate different sensitivities, even though the required precision in input may be roughly similar to our case study.

## B. CASE STUDY OF BOTTOM-REFLECTED RAYS USING A PARABOLIC BOTTOM

The preceding section presented a study of the accuracy of IMPULSE for rays which do not hit the boundaries, but, instead, are totally refracted. Another important question of accuracy concerns bottom-reflected paths. A convenient environment for studying this class of problems is one in which rays are straight line segments while the bottom forms a parabolic mirror with the source at the focus. In this case, all rays hitting the bottom are reflected to travel horizontally (in the analytic solution). We develop formulas for transmission loss in this case, compare numerical results with analytic solutions, and study the angles of the reflected rays.

Let the bottom depth $z_B$, as a function of range, be

$$z_B(x) = z_o + (a^2 + 2ax)^{1/2} , \qquad (19)$$

so that

$$x(z_B) = [(z_B - z_o)^2 - a^2]/2a. \qquad (20)$$

Equation (20) represents a parabola with a focus at range x = 0 and depth $z_o$.

To calculate the transmision loss for bottom-bounce rays, it is necessary to obtain a relation between the infinitesimal change in the launch angle and the infinitesimal change in the ray's depth at the receiver's range. Figure 9 shows a construction useful in obtaining the desired relation. Let X and Y be the points of intersection of the rays launched at $\theta_s$ and $\theta_s + \Delta\theta_s$, respectively, with the bottom. Then, with $\theta_B$ being the bottom angle,

$$\overline{XY} \sin (\theta_s - \theta_B) = \overline{SR}\, \Delta\theta_s \ , \tag{21}$$

$$\overline{XY} \sin \theta_B = \Delta z,$$

where $\overline{SR}$ is the "slant range," or distance from the source to X. Since $\Delta\theta_s$ is very small, the value of SR is the same to point Y as to X within terms of order $\Delta\theta_s$. We can ignore the higher order correction.

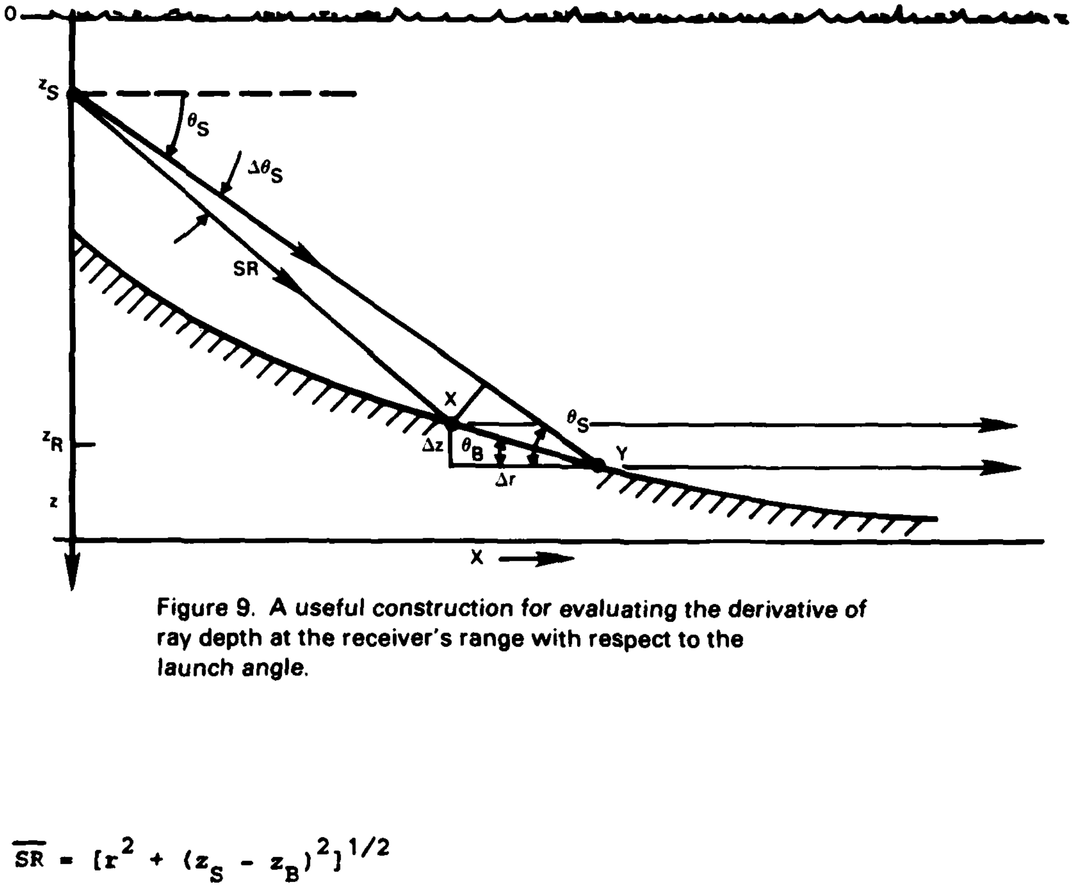


Figure 9. A useful construction for evaluating the derivative of ray depth at the receiver's range with respect to the launch angle.

Then

$$\overline{SR} = [r^2 + (z_S - z_B)^2]^{1/2}$$

or

$$\overline{SR} = r + a \qquad (22)$$

where r is the range of the point where the ray hits the bottom.

From Equations (21) and (22),

$$dz/d\theta_S = (r + a) \sin \theta_B / \sin (\theta_S - \theta_B). \qquad (23)$$

To evaluate Equation (23), values are needed for r and $\theta_B$. The range is found directly from Equation (19) for the bottom depth as a function of range which can be inverted for range as a function of depth (Eq. 20)). $\theta_B$ is found by differentiating the expression for depth with respect to range. Since the rays become horizontal after reflection, the bottom depth at the point of intersection is the same as the receiver depth, and, hence, at point X, the range is

$$x = [(z_R - z_S)^2 - a^2]/2a. \qquad (24)$$

Finally we have

$$dz_B/dx = \tan \theta_B = a/[a^2 + 2ax]^{1/2}$$

or

$$\sin \theta_B = a/[2a^2 + 2ax]^{1/2} . \qquad (25)$$

To evaluate the transmission loss, $\theta_S$ is also needed and is obtained from $\tan \theta_S = (z_B - z_S)/r$.

For our numerical study we choose $z_o = 1000$, $z_R = 3000$, $R = 5000$, $a = 1000$. An example of the bottom profile and the ray paths for these parameters is given in Figure 10. As in the Hirsch SPP study, we have used a special routine TESTB that creates a data file for use by IMPULSE which requires only a small number of input data such as NPTS2 (the number of bottom profile data to be created), a, DXBOT, $z_S$, $z_R$, DX, R, $\theta 1$, $D\theta$, $\theta 2$, and plotting switches. A listing of TESTB is presented in Appendix B.

For this example, the range at which the eigenray strikes the bottom is 1500. The bottom slope at this point is 0.5°, and, hence, $\theta_B = 26.57°$. The eigenray leaves the source with slope 1.333, and, hence, $\theta_S = 53.13°$. The transmission loss is TL = 73.2 dB.

Figure 11 shows a comparison between numerical results and the analytic result for TL as a function of the number of points used to define the bottom profile.

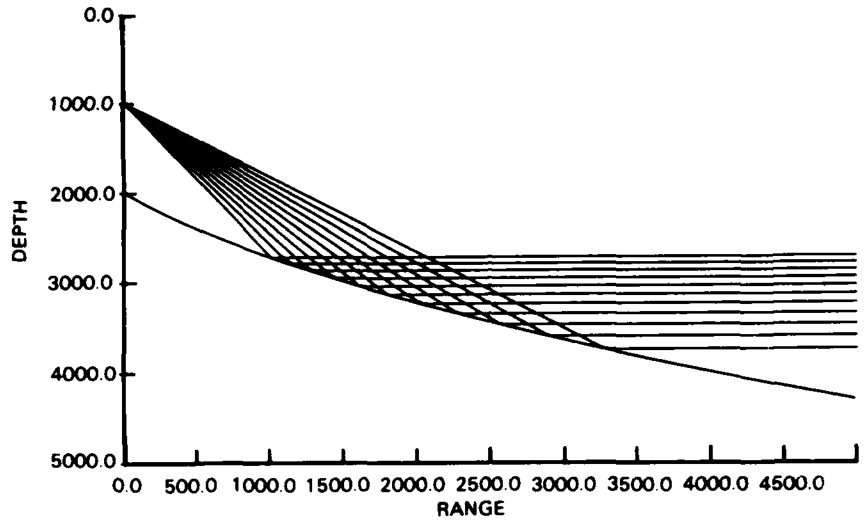


Figure 10. Ray paths and bottom depth are shown for a parabolic-mirror bottom. For these results, NPT2 = 50, a = 1000, DXBOT = 1, ZS = 1000, DX = 50, $\theta_1$ = 40°, D$\theta$ = 2°, and $\theta_2$ = 60°.

Figure 12 shows the scatter in ray angles reflected from the bottom and as a function of NPTS2. In both the TL and angle calculations, convergence is relatively good for NPTS2 greater than 25.

## C. CASE STUDY USING A REPRESENTATIVE SOUND SPEED PROFILE IN THE NORTH PACIFIC OCEAN

This section presents a typical example of the use of IMPULSE to calculate the impulse response for a bottom-mounted receiver and a near-surface source in an SSP typical of the North Pacific Ocean. Also included are a (hypothetical) bottom loss function and bottom-depth function. The primary aim here is not to obtain extremely high accuracy, but only to demonstrate, for a specific numerical case, how one would apply the general procedure for using IMPULSE as discussed in Section IID. This includes typical problems often encountered in practice such as obtaining reasonable fits to the SSP and locating eigenrays.

This example calculation shows that an uncertainty in the end results is unavoidable because of the finite number of environmental data, and the ensuing freedom allowed in generating reasonable curves to interpolate between the given data. A personal judgement must, ultimately, be introduced in this area. The example also shows a characteristic of long-range undersea ray paths, namely great sensitivity of the endpoint depth and angle to the initial angle.

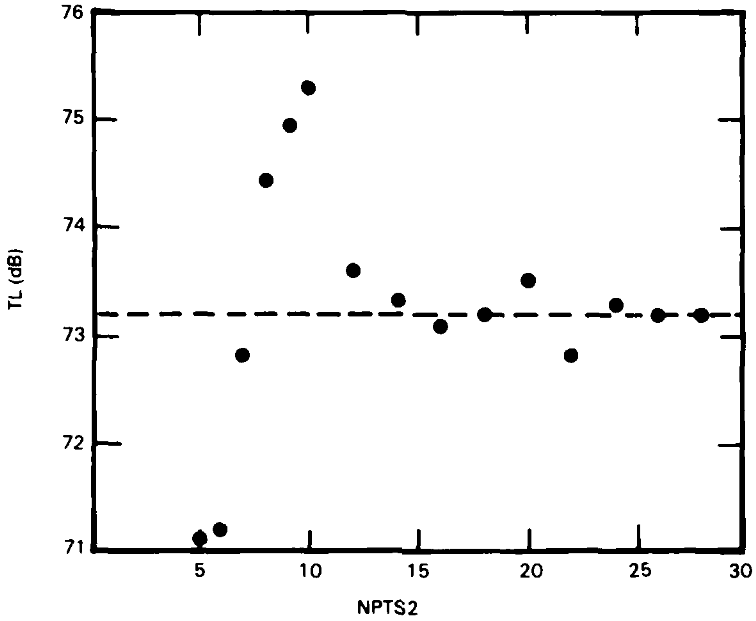


Figure 11. For the parabolic mirror bottom, the transmission loss from IMPULSE is compared to the analytic result (dashed line) for different numbers of bottom depth data NPTS2. In these runs, DX = 200, DTHETA = 1°, R = 5000, $z_S$ = 1000, $z_R$ = 3000, DXBOT = 10, a = 1000, and the sound speed is constant.

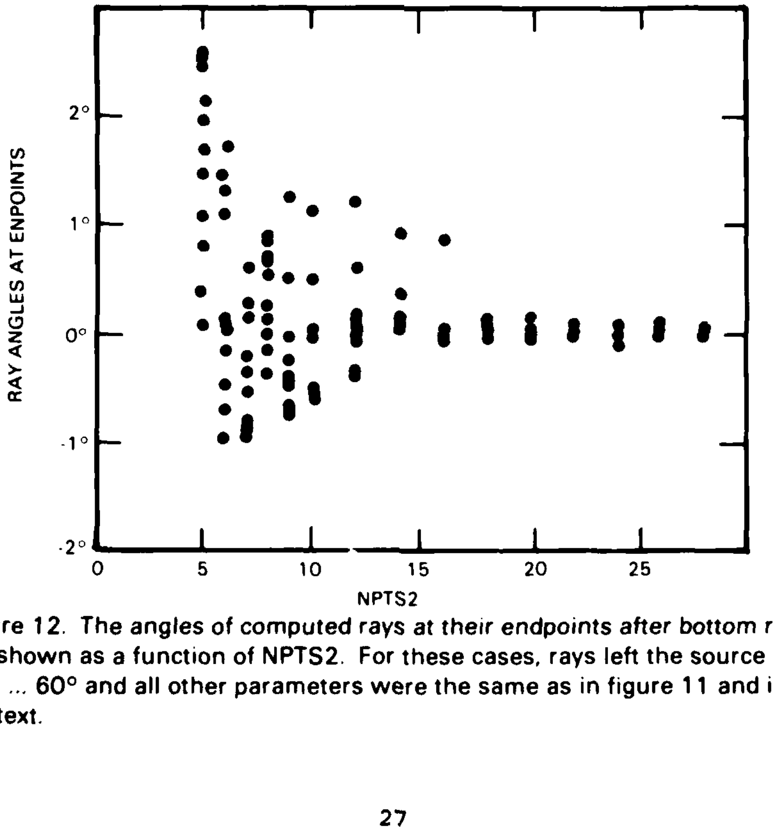
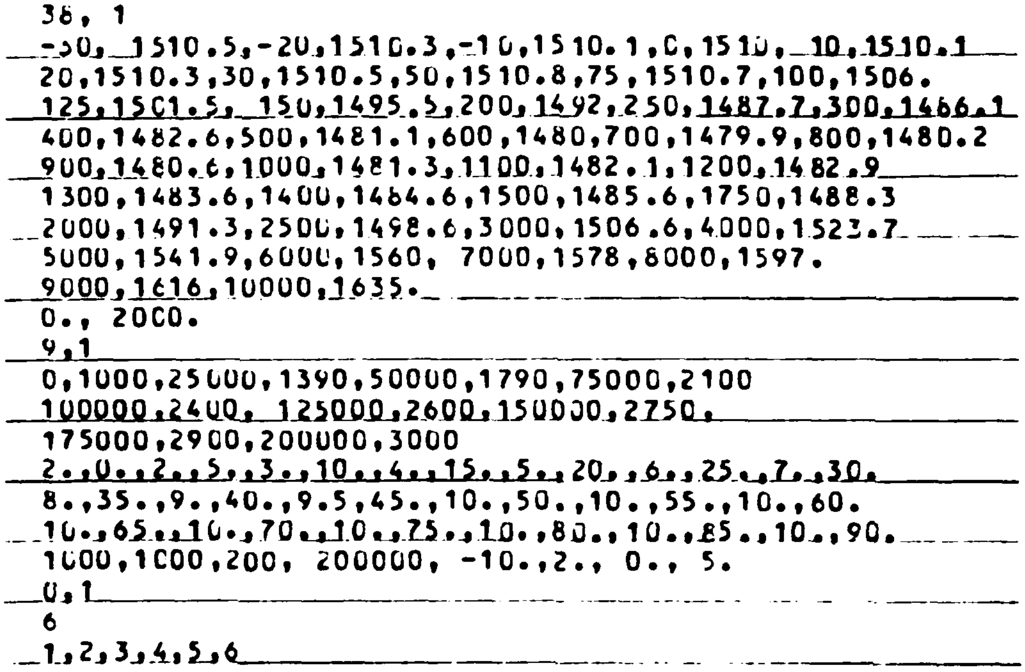


Figure 12. The angles of computed rays at their endpoints after bottom reflection are shown as a function of NPTS2. For these cases, rays left the source at 50°, 51°, ... 60° and all other parameters were the same as in figure 11 and in the text.

The first step is to examine the environmental data and, if necessary, add or delete data so as to obtain reasonable curve fits to the data. The SSP chosen for this case study is from the Colborn-Wright data base (references 7,8), province 19, Fall, in the North Pacific Ocean. Figure 13 shows the input data for a trial run. There are 38 SSP data pairs. The SSP plot data is dumped (onto external unit number 9) between 0 and 1000 m. A set of 19 bottom depths is specified out to range 200 km. The bottom loss is purely hypothetical. The source depth is 1000 m, the same as the bottom depth, a zero range. The range step size is 1000 m and the receiver is at a depth of 2000 m and range of 200 km. Therefore, in this case, rays actually are traced from receiver to source and reciprocity invoked. A set of six rays is traced from -10° to 0° in 2° steps. Bottom reflections are found to within 5 m in range. No printed trajectories are requested. Plots are requested for all six rays.

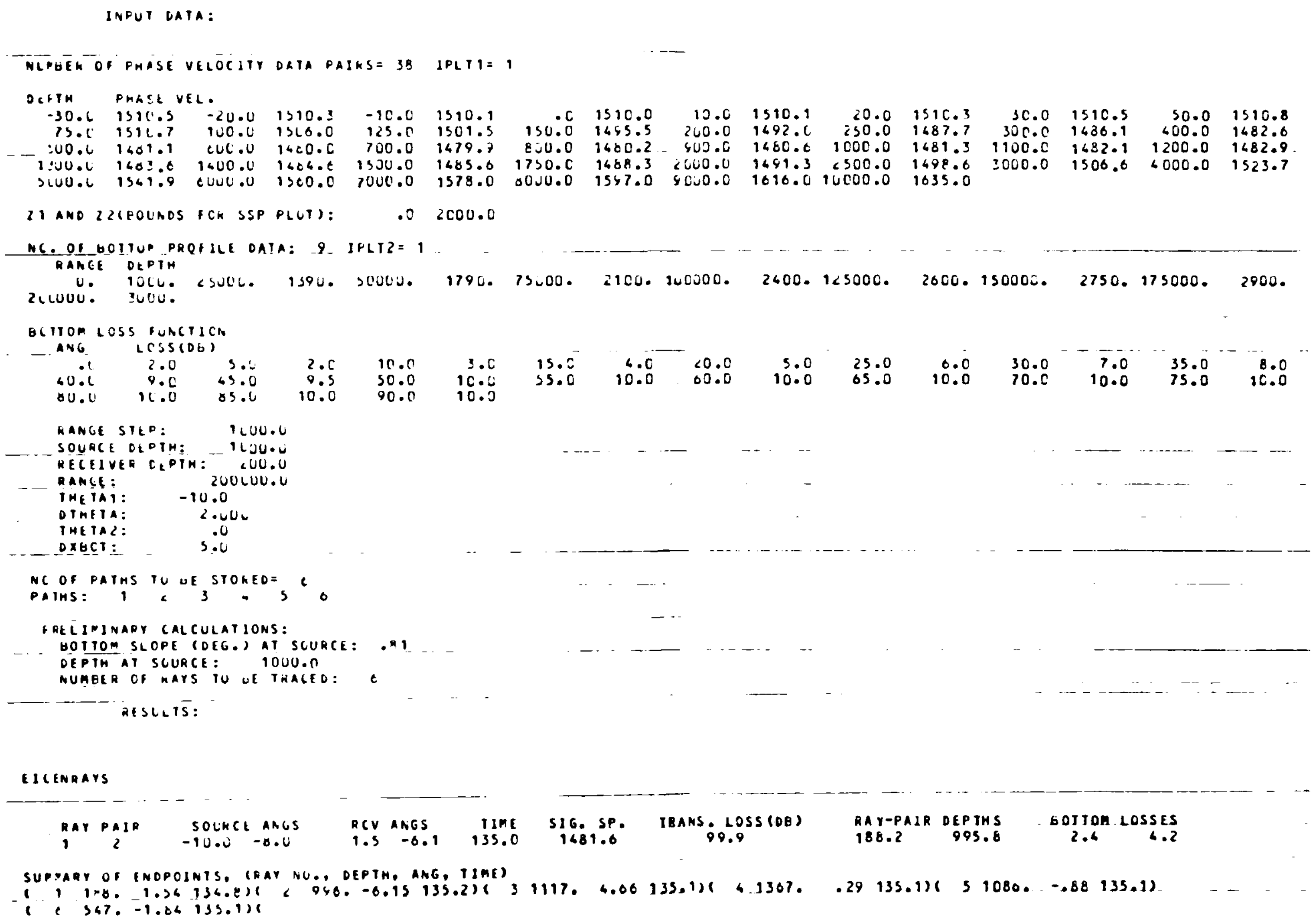


```
38, 1
-30,1510.5,-20,1510.3,-10,1510.1,0,1510,10,1510.1
20,1510.3,30,1510.5,50,1510.8,75,1510.7,100,1506.
125,1501.5, 150,1495.5,200,1492,250,1487.7,300,1486.1
400,1482.6,500,1481.1,600,1480,700,1479.9,800,1480.2
900,1480.6,1000,1481.3,1100,1482.1,1200,1482.9
1300,1483.6,1400,1484.6,1500,1485.6,1750,1488.3
2000,1491.3,2500,1498.6,3000,1506.6,4000,1523.7
5000,1541.9,6000,1560, 7000,1578,8000,1597.
9000,1616,10000,1635.
0., 2000.
9,1
0,1000,25000,1390,50000,1790,75000,2100
100000,2400, 125000,2600,150000,2750,
175000,2900,200000,3000
2.,0.,2.,5.,3.,10.,4.,15.,5.,20.,6.,25.,7.,30.
8.,35.,9.,40.,9.5,45.,10.,50.,10.,55.,10.,60.
10.,65.,10.,70.,10.,75.,10.,80.,10.,85.,10.,90.
1000,1000,200, 200000, -10.,2., 0., 5.
0,1
6
1,2,3,4,5,6
```

Figure 13. The input data are shown for the first trial run of IMPULSE using an SSP representative of the North Pacific Ocean.

As a point of interest, note that, before this run was made, the necessary file for storing plotting data was created and the external unit number 9 attached to it. The input data was stored on a file with external unit number 8 and IMPULSE was modified so as to read from unit 8 rather than from 5 (the card reader).

Figure 14 shows the printed output from IMPULSE for the trial run. Although an eigenray pair is found, the calculation is not at a very high precision. It is to be expected that further runs with greater precision (smaller range step) will move the rays to new positions.



```
          INPUT DATA:


 NUMBER OF PHASE VELOCITY DATA PAIRS= 38  IPLT1= 1

 DEPTH     PHASE VEL.
   -30.0  1510.5   -20.0  1510.3   -10.0  1510.1      .0  1510.0    10.0  1510.1    20.0  1510.3    30.0  1510.5    50.0  1510.8
    75.0  1510.7   100.0  1506.0   125.0  1501.5   150.0  1495.5   200.0  1492.0   250.0  1487.7   300.0  1486.1   400.0  1482.6
   500.0  1481.1   600.0  1480.0   700.0  1479.9   800.0  1480.2   900.0  1480.6  1000.0  1481.3  1100.0  1482.1  1200.0  1482.9
  1300.0  1483.6  1400.0  1484.6  1500.0  1485.6  1750.0  1488.3  2000.0  1491.3  2500.0  1498.6  3000.0  1506.6  4000.0  1523.7
  5000.0  1541.9  6000.0  1560.0  7000.0  1578.0  8000.0  1597.0  9000.0  1616.0 10000.0  1635.0

 Z1 AND Z2(BOUNDS FOR SSP PLOT):      .0  2000.0

 NO. OF BOTTOM PROFILE DATA:  9  IPLT2= 1
    RANGE  DEPTH
      0.   1000.  25000.   1390.  50000.   1790.  75000.   2100. 100000.   2400. 125000.   2600. 150000.   2750. 175000.   2900.
 200000.   3000.

 BOTTOM LOSS FUNCTION
    ANG      LOSS(DB)
      .0      2.0     5.0     2.0    10.0     3.0    15.0     4.0    20.0     5.0    25.0     6.0    30.0     7.0    35.0     8.0
    40.0      9.0    45.0     9.5    50.0    10.0    55.0    10.0    60.0    10.0    65.0    10.0    70.0    10.0    75.0    10.0
    80.0     10.0    85.0    10.0    90.0    10.0

    RANGE STEP:       1000.0
    SOURCE DEPTH:     1000.0
    RECEIVER DEPTH:    200.0
    RANGE:          200000.0
    THETA1:      -10.0
    DTHETA:        2.000
    THETA2:         .0
    DXBCT:         5.0

 NO OF PATHS TO BE STORED=  6
 PATHS:   1   2   3   4   5   6

  PRELIMINARY CALCULATIONS:
    BOTTOM SLOPE (DEG.) AT SOURCE:  .81
    DEPTH AT SOURCE:     1000.0
    NUMBER OF RAYS TO BE TRACED:   6

           RESULTS:


 EIGENRAYS

    RAY PAIR      SOURCE ANGS       RCV ANGS     TIME   SIG. SP.   TRANS. LOSS(DB)      RAY-PAIR DEPTHS    BOTTOM LOSSES
    1    2       -10.0   -8.0      1.5  -6.1   135.0    1481.6          99.9            188.2      995.8      2.4      4.2

 SUMMARY OF ENDPOINTS, (RAY NO., DEPTH, ANG, TIME)
 (  1  188.  -1.54 134.8)(  2  996. -6.15 135.2)(  3 1117.  4.66 135.1)(  4 1367.   .29 135.1)(  5 1086.  -.88 135.1)
 (  6  547. -1.64 135.1)(
```

Figure 14. The printout from IMPULSE is shown for the test case input in figure 13.

Figure 15 shows the SSP plot from PLOTS, the plotting adjunct which uses the output file on unit 9 as input. (This file is attached to unit 10 before running PLOTS.) The curve fit is unsatisfactory because of the spurious local maximum at about 200 m depth. There is no unique procedure for generating a better SSP data set and one must use personal judgement in adding or deleting data.

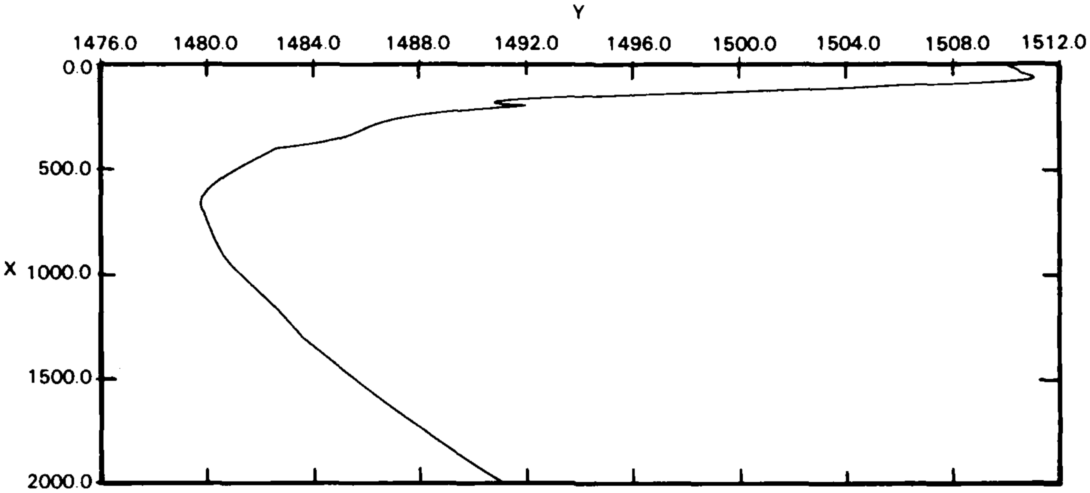


Figure 15. The SSP as fit by the piece-wise continuous cubics of IMPULSE. Note the spurious local maximum near the *200 m depth.*

The solution of adding data was employed in this test case. On a separate plot (not shown), the initial SSP data were plotted to a depth of 500 m. Then a smooth curve was sketched between data. Where Figure 15 shows unreasonable curvature, new data, lying on this interpolated curve, were inserted. The input data file was augmented with these data at depths of 40, 175, 225, 275, 350, and 450 m. Figure 16 shows the new data file and figure 17, the new SSP, which is satisfactory for present purposes.

A procedure exactly analogous to that for the SSP data revision can be applied to smooth any undesirable curve-fitting artifacts in the bottom depth function. For this test case, it turns out that the curve fit is reasonably free of undesirable features.

The revised printout from IMPULSE is shown in Figure 18 — the eigenrays are gone.

The next step is to locate eigenrays. Figure 19 shows the ray paths obtained from the second run with a revised SSP.

As a point of interest, the user should be aware that plots showing rays reflecting from the bottom often will not show the actual point of contact. In other words, the ray will appear to bounce from a point above the bottom. This does not imply errors have occurred in the ray-tracing or plotting routines. It actually arises because the plots of ray paths are made from points which are spaced by DX in range. The plotting routine, of course, has no means of knowing where the ray went between these points. Consequently, two points before, and after, a bottom reflection will just be connected directly by the plotting software, without any line segment touching the bottom. The same artifact will also be seen for surface reflections.

The sound speed at the source, and at the receiver, are useful in suggesting where eigenrays may lie. From Snell's law, the ratio of sound speeds can be used to determine a limiting ray angle, at the source, which must be exceeded simply in order for the ray to reach the receiver. In our case, the sound speed at the source is 1481.3 m/s and at the receiver, 1492 m/s, yielding a limiting ray angle of approximately 7°. Since the bottom slope at the source is roughly 1°, any eigenrays must leave the source in the upward direction. Therefore, as an initial fan of rays suggests eigenray locations, we take initial and final angles of -30° and -5°, respectively, and a step size of 5° between rays. The range step size is chosen to be 100 m, corresponding to 2000 steps from source to receiver. Figure 20 shows the ray paths and Figure 21 the printout for this case. Three eigenray pairs or flux-tubes are identified.

There is no guarantee that the eigenrays identified at this stage are valid, because the step size has not been reduced to the point where further reduction causes no change. In a set of tests (not shown), we find results stabilize for values of DX around 10 m. This is a remarkably small value, corresponding to 20,000 steps from source to receiver. In general, each environment must be evaluated in its own context and some cases will not require as many calculations. But, for bottom profiles with sharp angles or other special cases, one can expect extreme sensitivity of ray endpoints to small changes in launch angle.

The remaining flux tubes, after results have converged, are clustered around -24.8° and -24.4° at the source. Figure 22 shows this in printout form. For these rays, it is not desirable to obtain plots since such a huge number of points would be involved with the existing routines. It would be more straightforward to modify the plot routine so that each path would be described by 200 points (e.g., by throwing out points), but this has not, as yet, been accomplished.

```
44, 1
-30, 1510.5,-20,1510.3,-10,1510.1,0,1510, 10,1510.1
20,1510.3,30,1510.5,40,1510.65,50,1510.8,75,1510.7,100,1506
125,1501.5,150,1495.5,175,1493.75,200,1492,225,1489.85
250,1487.7,275,1486.9,300,1486.1,350,1484.35,400,1482.6
450,1481.85
500,1481.1,600,1480,700,1479.9,800,1480.2
900,1480.6,1000,1481.4,1100,1482.1,1200,1482.9
1300,1483.6,1400,1484.6,1500,1485.6,1750,1488.3
2000,1491.3,2500,1498.6,3000,1506.6,4000,1523.7
5000,1541.9,6000,1560, 7000,1578,8000,1597.
9000,1616,10000,1635.
0., 2000.
9,1
0,1000,25000,1390,50000,1790,75000,2100
100000,2400, 125000,2600,150000,2750,
175000,2900,200000,3000
2.,0.,2.,5.,3.,10.,4.,15.,5.,20.,6.,25.,7.,30.
8.,35.,9.,40.,9.5,45.,10.,50.,10.,55.,10.,60.
10.,65.,10.,70.,10.,75.,10.,80.,10.,85.,10.,90.
1000,1000,200, 200000, -10.,2., 0., 5.
0,1
6
1,2,3,4,5,6
```

Figure 16. This revised input shows additional SSP data at 40, 175, 225, 275, 350 and 450 m that forces the cubic fit closer to a reasonable interpolated curve between the initial SSP data.

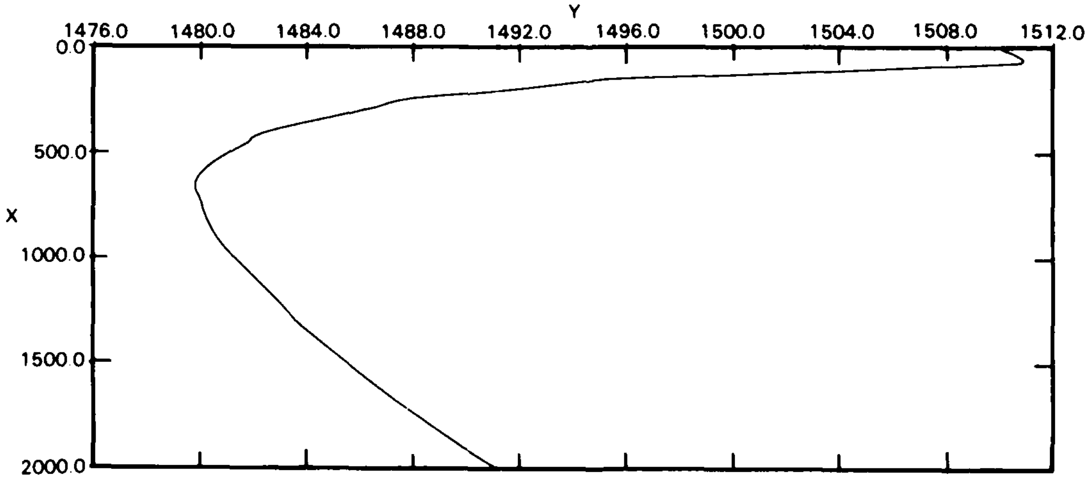
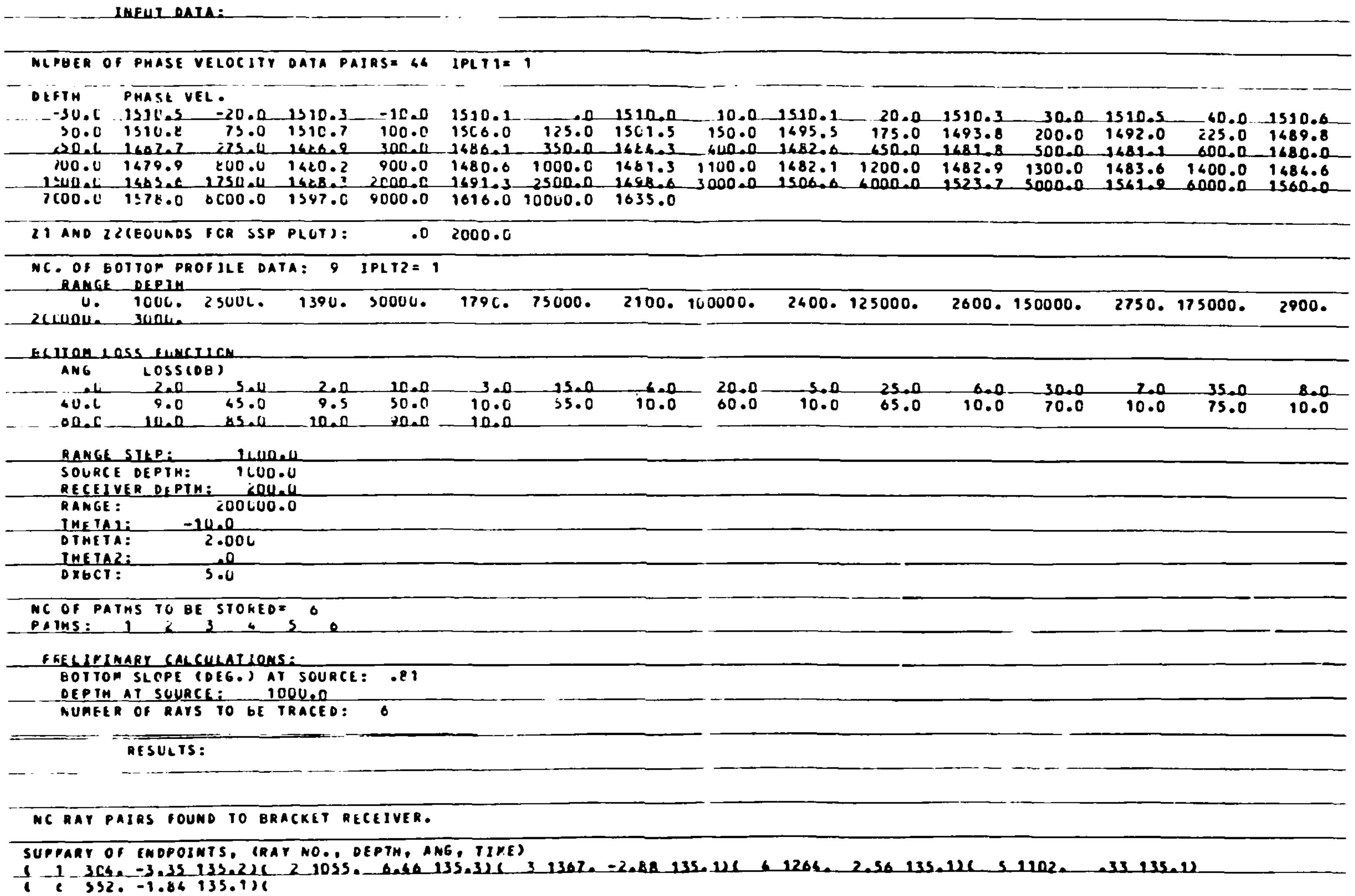


Figure 17. The revised SSP curve fit has no spurious local maxima and is adequate for present purposes.



```
          INPUT DATA:

NUMBER OF PHASE VELOCITY DATA PAIRS= 44  IPLT1= 1

DEPTH     PHASE VEL.
  -30.0  1510.5   -20.0  1510.3   -10.0  1510.1      .0  1510.0    10.0  1510.1    20.0  1510.3    30.0  1510.5    40.0  1510.6
   50.0  1510.8    75.0  1510.7   100.0  1506.0   125.0  1501.5   150.0  1495.5   175.0  1493.8   200.0  1492.0   225.0  1489.8
  250.0  1487.7   275.0  1486.9   300.0  1486.1   350.0  1484.3   400.0  1482.6   450.0  1481.8   500.0  1481.1   600.0  1480.0
  700.0  1479.9   800.0  1480.2   900.0  1480.6  1000.0  1481.3  1100.0  1482.1  1200.0  1482.9  1300.0  1483.6  1400.0  1484.6
 1500.0  1485.6  1750.0  1488.3  2000.0  1491.3  2500.0  1498.6  3000.0  1506.6  4000.0  1523.7  5000.0  1541.9  6000.0  1560.0
 7000.0  1578.0  8000.0  1597.0  9000.0  1616.0 10000.0  1635.0

Z1 AND Z2(BOUNDS FOR SSP PLOT):      .0  2000.0

NO. OF BOTTOM PROFILE DATA:  9  IPLT2= 1
   RANGE  DEPTH
     0.   1000.  25000.   1390.  50000.   1790.  75000.   2100. 100000.   2400. 125000.   2600. 150000.   2750. 175000.   2900.
200000.   3000.

BOTTOM LOSS FUNCTION
   ANG      LOSS(DB)
    .0     2.0     5.0     2.0    10.0     3.0    15.0     4.0    20.0     5.0    25.0     6.0    30.0     7.0    35.0     8.0
   40.0     9.0    45.0     9.5    50.0    10.0    55.0    10.0    60.0    10.0    65.0    10.0    70.0    10.0    75.0    10.0
   80.0    10.0    85.0    10.0    90.0    10.0

   RANGE STEP:       1000.0
   SOURCE DEPTH:     1000.0
   RECEIVER DEPTH:    200.0
   RANGE:          200000.0
   THETA1:       -10.0
   DTHETA:         2.000
   THETA2:          .0
   DXBCT:          5.0

NO OF PATHS TO BE STORED=  6
PATHS:   1   2   3   4   5   6

  PRELIMINARY CALCULATIONS:
   BOTTOM SLOPE (DEG.) AT SOURCE:  .81
   DEPTH AT SOURCE:     1000.0
   NUMBER OF RAYS TO BE TRACED:   6

          RESULTS:

 NO RAY PAIRS FOUND TO BRACKET RECEIVER.

SUMMARY OF ENDPOINTS, (RAY NO., DEPTH, ANG, TIME)
(  1  304. -3.35 135.2)(  2 1055.  6.46 135.3)(  3 1367. -2.88 135.1)(  4 1264.  2.56 135.1)(  5 1102.   .33 135.1)
(  6  552. -1.84 135.1)(
```

Figure 18. The revised IMPULSE printout. Note that no eigenrays are found.

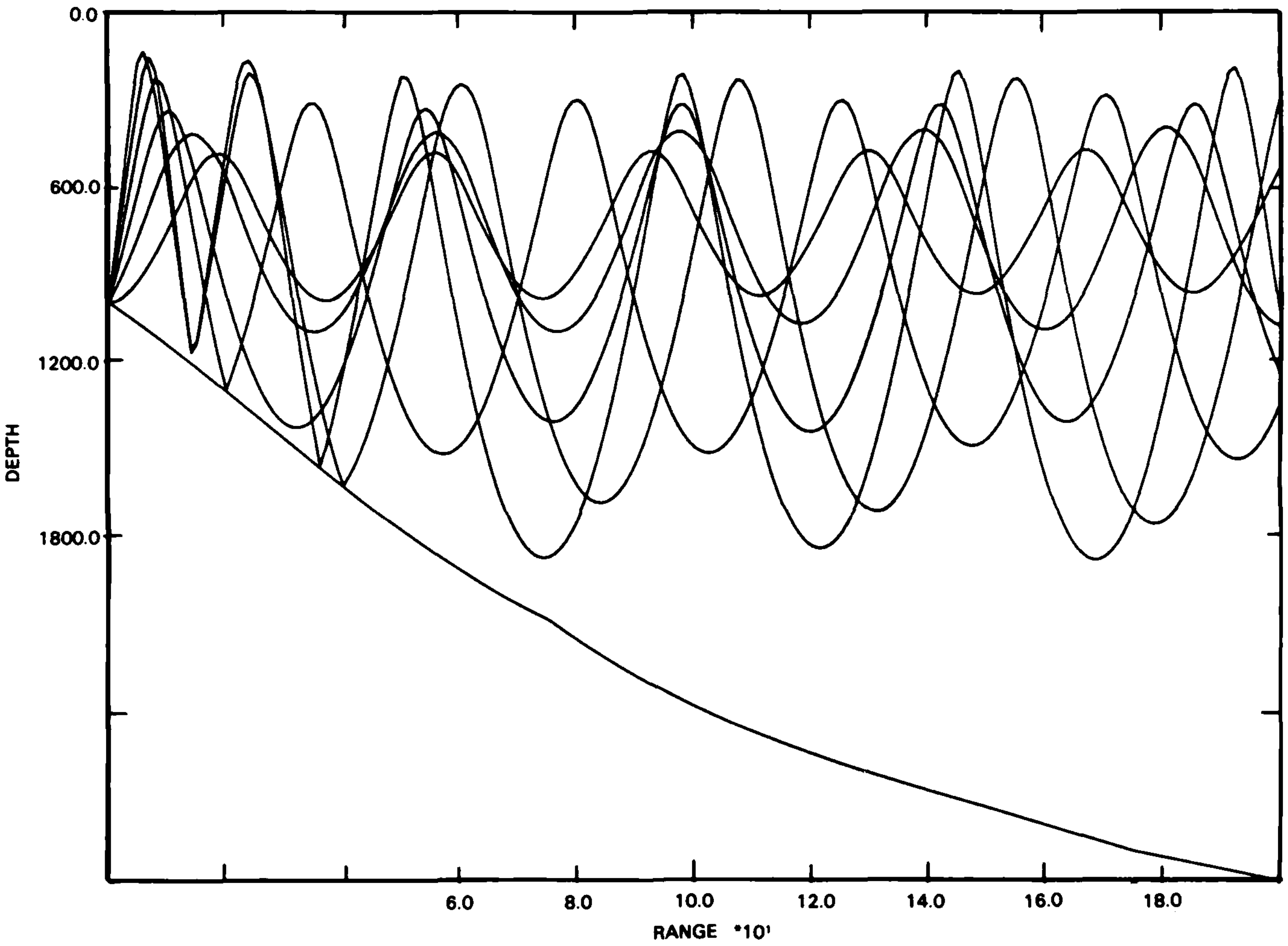


Figure 19. Ray paths and bottom profile are shown for the test case — for launch angles from -10° to 0° in 2° steps. Bottom reflected rays may appear to reflect from points above the bottom.

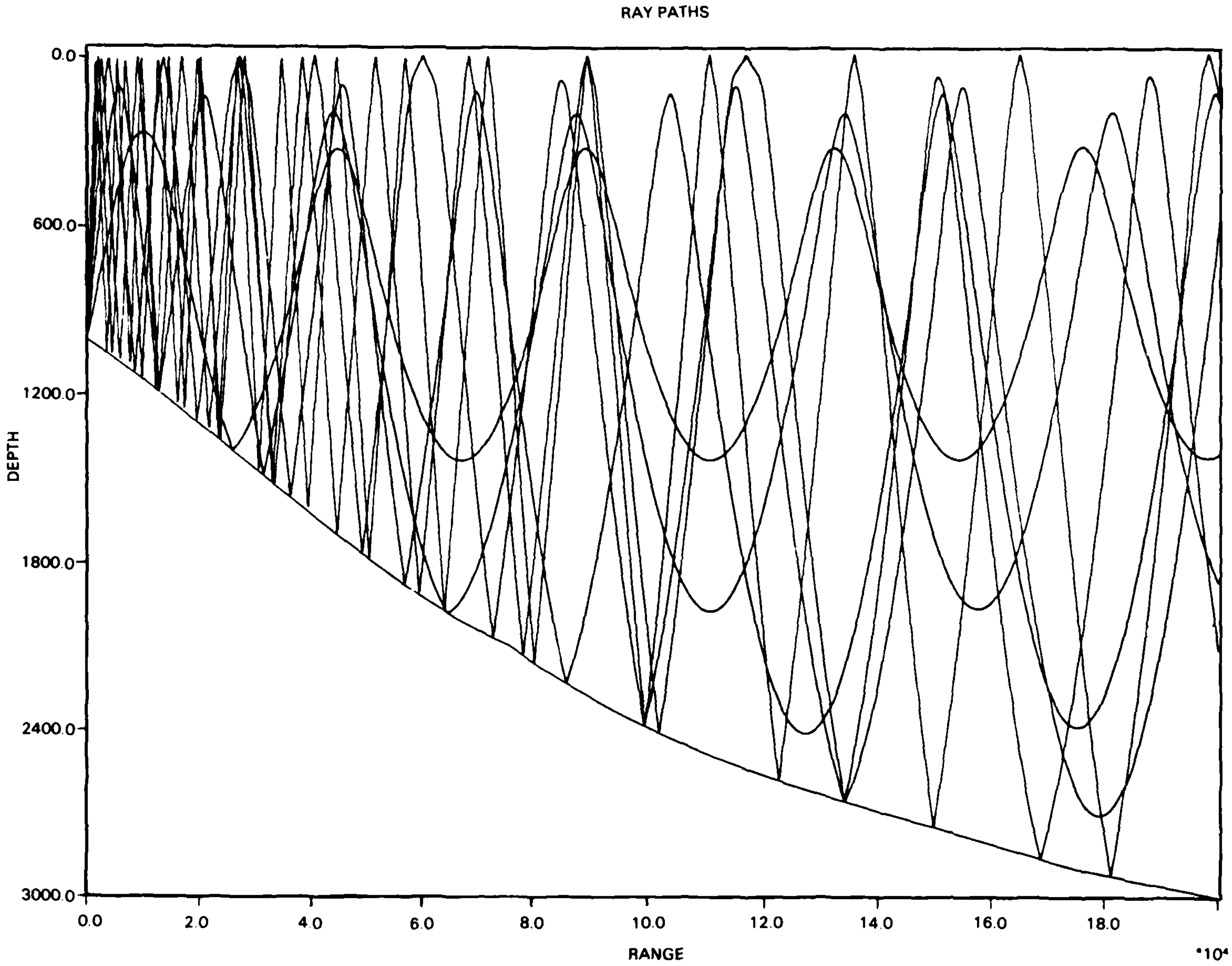
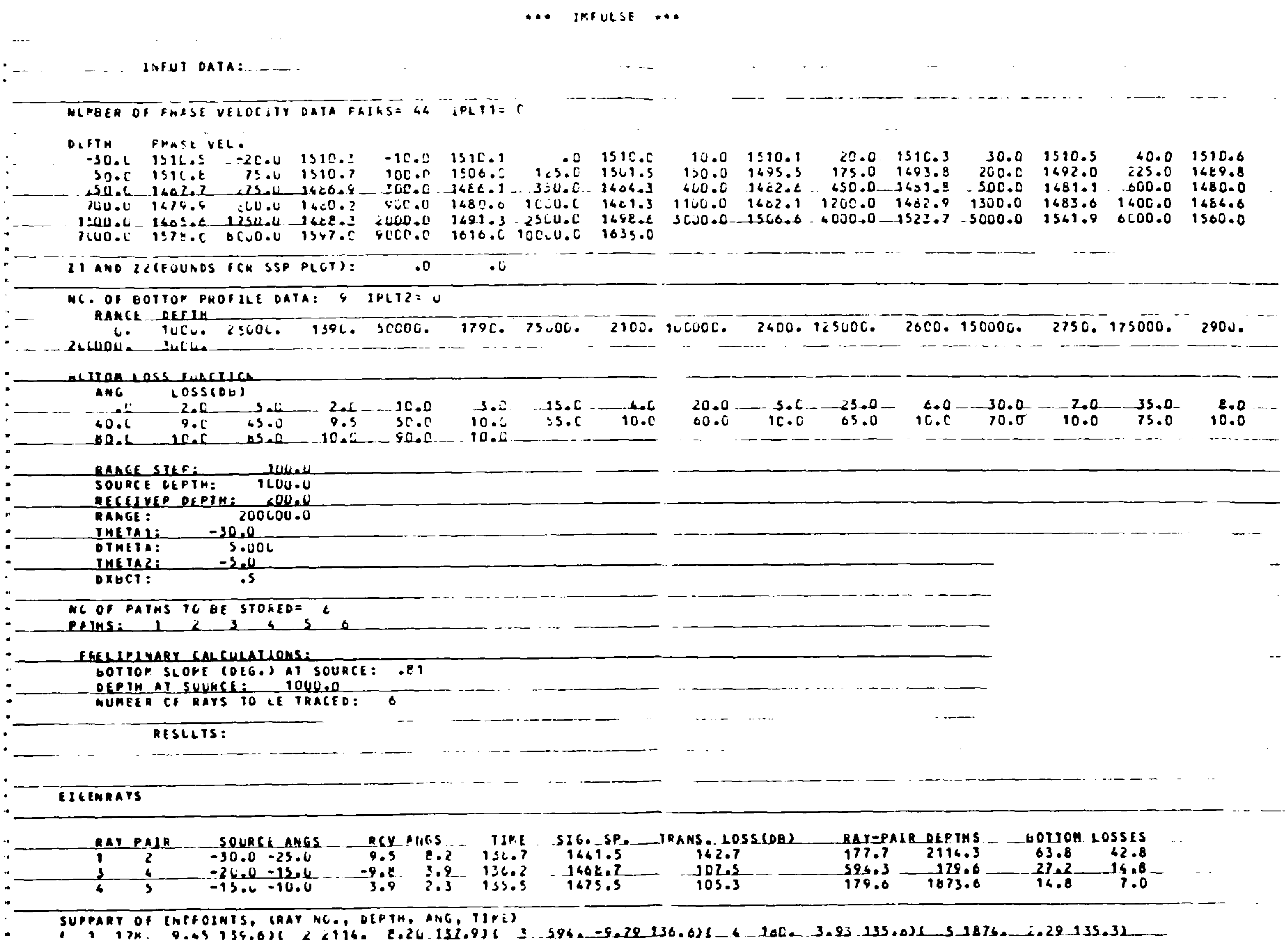
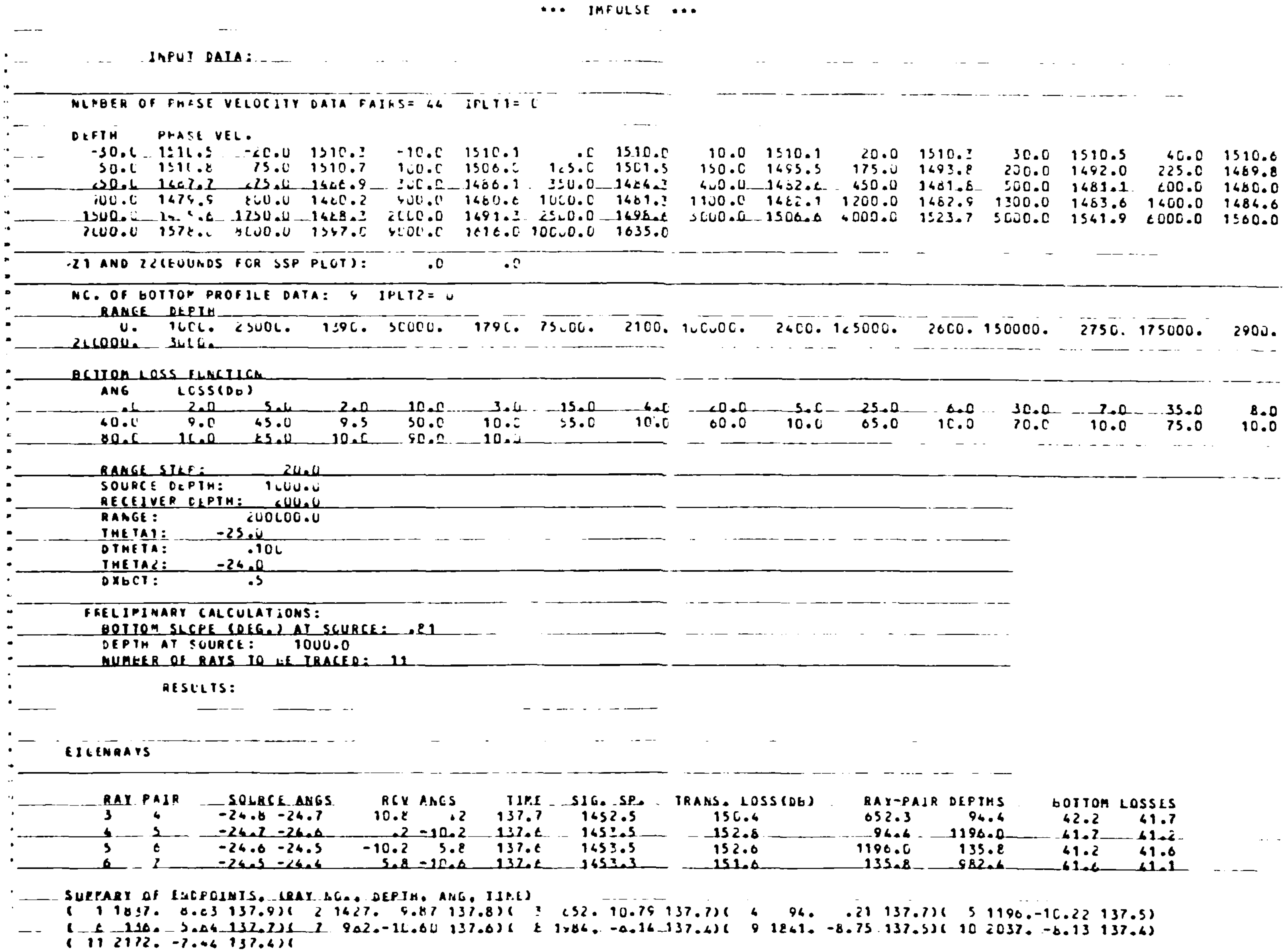


Figure 20. A fan of  rays is shown leaving the source between -30° and -5° in 5° steps.



```
                                   *** IMPULSE ***

         INPUT DATA:

   NUMBER OF PHASE VELOCITY DATA PAIRS= 44  IPLT1= 0

   DEPTH     PHASE VEL.
     -30.0   1510.5   -20.0  1510.3    -10.0  1510.1       .0  1510.0     10.0  1510.1     20.0  1510.3     30.0  1510.5     40.0  1510.6
      50.0   1510.8    75.0  1510.7    100.0  1506.0    125.0  1501.5    150.0  1495.5    175.0  1493.8    200.0  1492.0    225.0  1489.8
     250.0   1487.7   275.0  1486.9    300.0  1486.1    350.0  1484.3    400.0  1482.8    450.0  1481.5    500.0  1481.1    600.0  1480.0
     700.0   1479.9   800.0  1480.2    900.0  1480.6   1000.0  1481.3   1100.0  1482.1   1200.0  1482.9   1300.0  1483.6   1400.0  1484.6
    1500.0   1485.6  1750.0  1488.3   2000.0  1491.3   2500.0  1498.6   3000.0  1506.6   4000.0  1523.7   5000.0  1541.9   6000.0  1560.0
    7000.0   1578.0  8000.0  1597.0   9000.0  1616.0  10000.0  1635.0

   Z1 AND Z2(BOUNDS FOR SSP PLOT):        .0        .0

   NO. OF BOTTOM PROFILE DATA:  9  IPLT2= 0
      RANGE  DEPTH
        0.   1000.  25000.   1390.  50000.   1790.  75000.   2100. 100000.   2400. 125000.   2600. 150000.   2750. 175000.   2900.
   200000.   3000.

   BOTTOM LOSS FUNCTION
      ANG      LOSS(DB)
        .0      2.0     5.0     2.0     10.0     3.0     15.0     4.0     20.0     5.0     25.0     6.0     30.0     7.0     35.0     8.0
      40.0      9.0    45.0     9.5     50.0    10.0     55.0    10.0     60.0    10.0     65.0    10.0     70.0    10.0     75.0    10.0
      80.0     10.0    85.0    10.0     90.0    10.0

      RANGE STEP:          100.0
      SOURCE DEPTH:       1000.0
      RECEIVER DEPTH:      200.0
      RANGE:            200000.0
      THETA1:          -30.0
      DTHETA:            5.000
      THETA2:           -5.0
      DXBCT:              .5

   NO OF PATHS TO BE STORED=  6
   PATHS:   1   2   3   4   5   6

     PRELIMINARY CALCULATIONS:
      BOTTOM SLOPE (DEG.) AT SOURCE:   .81
      DEPTH AT SOURCE:     1000.0
      NUMBER OF RAYS TO BE TRACED:    6

            RESULTS:

  EIGENRAYS

      RAY PAIR      SOURCE ANGS      RCV ANGS      TIME    SIG. SP.   TRANS. LOSS(DB)     RAY-PAIR DEPTHS      BOTTOM LOSSES
       1    2      -30.0 -25.0      9.5   8.2    136.7     1441.5        142.7           177.7    2114.3        63.8      42.8
       3    4      -20.0 -15.0     -9.8   3.9    136.2     1468.7        107.5           594.3     179.6        27.2      14.8
       4    5      -15.0 -10.0      3.9   2.3    135.5     1475.5        105.3           179.6    1873.6        14.8       7.0

  SUMMARY OF ENDPOINTS, (RAY NO., DEPTH, ANG, TIME)
  ( 1  178.   9.45 136.6)( 2 2114.  8.20 137.9)( 3  594. -9.79 136.6)( 4  180.  3.93 135.6)( 5 1874.  2.29 135.3)
```

Figure 21. The printout for a fan of rays from -30° to -5° in 5° steps.



```
                                        ***  IMPULSE  ***

          INPUT DATA:

     NUMBER OF PHASE VELOCITY DATA PAIRS= 44   IPLT1= 0

     DEPTH     PHASE VEL.
        -30.0  1510.5   -20.0  1510.3   -10.0  1510.1      .0  1510.0    10.0  1510.1    20.0  1510.3    30.0  1510.5    40.0  1510.6
         50.0  1510.8    75.0  1510.7   100.0  1506.0   125.0  1501.5   150.0  1495.5   175.0  1493.8   200.0  1492.0   225.0  1489.8
        250.0  1487.7   275.0  1486.9   300.0  1486.1   350.0  1484.3   400.0  1482.6   450.0  1481.8   500.0  1481.1   600.0  1480.0
        700.0  1479.9   800.0  1480.2   900.0  1480.6  1000.0  1481.3  1100.0  1482.1  1200.0  1482.9  1300.0  1483.6  1400.0  1484.6
       1500.0  1485.6  1750.0  1488.3  2000.0  1491.3  2500.0  1496.6  3000.0  1506.6  4000.0  1523.7  5000.0  1541.9  6000.0  1560.0
       7000.0  1578.[illegible]  8000.0  1597.0  9000.0  1616.0 10000.0  1635.0

     Z1 AND Z2(BOUNDS FOR SSP PLOT):        .0        .0

     NO. OF BOTTOM PROFILE DATA:   9   IPLT2= 0
        RANGE   DEPTH
          0.    1000.  25000.   1390.  50000.   1790.  75000.   2100. 100000.   2400. 125000.   2600. 150000.   2750. 175000.   2900.
     200000.   3000.

     BOTTOM LOSS FUNCTION
        ANG      LOSS(DB)
          .0      2.0     5.0     2.0    10.0     3.0    15.0     4.0    20.0     5.0    25.0     6.0    30.0     7.0    35.0     8.0
        40.0      9.0    45.0     9.5    50.0    10.0    55.0    10.0    60.0    10.0    65.0    10.0    70.0    10.0    75.0    10.0
        80.0     10.0    85.0    10.0    90.0    10.0

        RANGE STEP:         20.0
        SOURCE DEPTH:     1000.0
        RECEIVER DEPTH:    600.0
        RANGE:          200000.0
        THETA1:       -25.0
        DTHETA:          .100
        THETA2:       -24.0
        DXBCT:           .5

      PRELIMINARY CALCULATIONS:
        BOTTOM SLOPE (DEG.) AT SOURCE:  .81
        DEPTH AT SOURCE:     1000.0
        NUMBER OF RAYS TO BE TRACED:  11

                RESULTS:


     EIGENRAYS

        RAY PAIR     SOURCE ANGS     RCV ANGS      TIME    SIG. SP.   TRANS. LOSS(DB)     RAY-PAIR DEPTHS     BOTTOM LOSSES
         3    4      -24.8 -24.7    10.8    .2    137.7    1452.5        150.4             652.3     94.4      42.2    41.7
         4    5      -24.7 -24.6      .2 -10.2    137.6    1453.5        152.8              94.4   1196.0      41.7    41.2
         5    6      -24.6 -24.5   -10.2   5.8    137.6    1453.5        152.6            1196.0    135.8      41.2    41.6
         6    7      -24.5 -24.4     5.8 -10.6    137.6    1453.3        151.6             135.8    982.4      41.6    41.1

     SUMMARY OF ENDPOINTS. (RAY NO., DEPTH, ANG, TIME)
     (  1 1837.   8.63 137.9)(  2 1427.   9.87 137.8)(  3  652. 10.79 137.7)(  4   94.   .21 137.7)(  5 1196.-10.22 137.5)
     (  6  136.   5.84 137.7)(  7  982.-10.60 137.6)(  8 1984.  -6.14 137.4)(  9 1841.  -8.75 137.5)( 10 2037.  -6.13 137.4)
     ( 11 2172.  -7.44 137.4)(
```

Figure 22. This printout shows a small range step size with a cluster of ray pair that bracket the  receiver depth.

# REFERENCES

1. Northrop, J., and Shockley, R.C., "Long-range Pacific Acoustic Multipath Identification," J. Acoust. Soc. AM. 84, p. 1760-1765, 1984

2. National Defense Research Committee, Dept. of the Navy, Physics of Sound in the Sea, v. 8, p. 53, 1969

3. Raleigh, J.W.S., The Theory of Sound, Dover Publications (NY), v. II, p. 145, 1945

4. Bucker, H. (To be published)

5. Abramowitz, M. and Stegun, I., eds., Handbook of Mathematical Functions, National Bureau of Standards, 9th printing, p. 897, 1970

6. Hirsch, P. and Carter, A.H., J. Acoust. Soc. Am. 37, p. 90, 1965

7. Ocean Data Systems, Inc., Final Technical Report, Pacific Ocean/ASEPS Sound-speed Profiles, Wright-analyzed Watermass Data Base, by J. Crouch and K. Osborn, July 1981

8. Ocean Data Systems, Inc., Final Technical Report, Atlantic Ocean/ASEPS Sound-speed Profiles, Colborn-analyzed Watermass Data Base, by J. Crouch and K. Osborn, July 1981

# APPENDIX A. LISTING OF IMPULSE

Symbolic listings of the main routine PULSE and subroutines FITSSP, FITBOT, FITRAY, FUNC, GETBL, PLTSSP, PLTBOT, PLOTS are shown below.

Main routine — PULSE

```
C     IMPULSE:  A FORTRAN CODE FOR CALCULATING THE IMPULSE RESPONSE FOR
C A BOTTOM-MOUNTED OR SUSPENDED SOURCE AT A SUSPENDED RECEIVER.  FROM
C THE RECIPROCITY THEOREM, THIS ALSO SUFFICES FOR CALCULATING THE IMPULSE
C RESPONSE AT A BOTTOM-MOUNTED RECEIVER FOR A SUSPENDED SOURCE.
C
C     THE CODE TREATS A SINGLE SOUND SPEED PROFILE AND A BOTTOM
C WHICH IS A SPECULAR REFLECTOR, HAS RANGE-VARYING DEPTH, AND A
C SINGLE BOTTOM LOSS FUNCTION.
C
C     THE CODE TRACES RAYS TO THE RECEIVER RANGE AT ANGULAR INCREMENTS
C SPECIFIED BY THE USER (DTHETA) BETWEEN INITIAL AND FINAL ANGLES (THETA1
C AND THETA2, WITH THETA2 GT THETA1).  POSITIVE DEPTH, ANGLES, AND SLOPES
C ARE ALL IN THE DOWNWARD DIRECTION.  UP TO 400 RAYS MAY BE TRACED AND 100
C EIGENRAYS FOUND.  TIME AND INTENSITY ARE COMPUTED FOR EACH EIGENRAY.
C
C     RAYS ARE FOUND BY USING A FOURTH-ORDER RUNGE-KUTTA SCHEME (REF 1)
C TO SOLVE THE RAY EQUATION OF MOTION:
C
C         D(DZ/DX)/DX = -(1+DZ/DX**2)*(1/C)*DC/DZ.
C
C ONCE THE RECEIVER RANGE IS REACHED, PAIRS OF RAYS THAT WERE ADJACENT
C AT THE SOURCE AND WHICH BRACKET (PASS ABOVE AND BELOW) THE RECEIVER
C ARE IDENTIFIED.  EACH SUCH PAIR IMPLY THE PRESENCE OF AN EIGENRAY LYING
C BETWEEN THEM.  THE TRANSMISSION LOSS FOR EACH PAIR IS CALCULATED ACCOR-
C DING TO THE GEOMETRICAL SPREADING FORMULA (REF 2):
C
C     TL = -10.*LOG10((1/R)*COS(THETAS)/(COS(THETAR)*(DZ/DTHETA))),
C
C WHERE THETAS (THETAR) IS THE ANGLE OF THE RAY AT THE SOURCE (RECEIVER),
C R IS THE HORIZONTAL RANGE FROM SOURCE TO RECEIVER, AND DZ/DTHETA THE
C DERIVATIVE OF DEPTH WITH RESPECT TO LAUNCH ANGLE, EVALUATED AT RANGE R.
C THIS DERIVATIVE IS ESTIMATED BY A RATIO OF THE DIFFERENCE IN THE DEPTH
C OF THE RAYS IN EACH PAIR TO THE ANGULAR DIFFERENCE AT THE SOURCE.
C HOW SENSITIVE THE RESULTS ARE TO INITIAL RAY DENSITY CAN BE
C STUDIED BY MAKING RUNS WITH SMALLER INCREMENTS IN INITIAL RAY ANGLES.
C TO SAVE STORAGE THE CODE ONLY RETAINS THE CURRENT VALUES OF THE
C RAY SLOPE, TIME, AND DEPTH (PLUS TWO PREVIOUS DEPTHS FOR USE IN
C FINDING THE INTERSECTION OF LOCAL RAY PATHS AND THE BOTTOM).
```

```
C     NO ATTEMPT IS MADE TO ESTIMATE THE SUM OF THE SEPARATE MULTIPATH
C CONTRIBUTIONS, AS THIS WOULD REQUIRE KNOWLEDGE OF THE PHASE ALONG EACH
C PATH, WHICH CALCULATION REQUIRES IDENTIFYING CAUSTICS.  COHERENT SUMMA-
C TION ALSO WOULD NOT FAIRLY REPRESENT THE FLUCTUATIONS PRESENT IN MOST
C MEDIA.  WE DO NOT ATTEMPT TO ACCOUNT FOR ENERGY THAT ARRIVES VIA
C EVANESCENT WAVES, THAT IS, BY OTHER THAN DIRECT PATH PROPAGATION.
C
C     BOTTOM REFLECTION LOSS IS INPUT BY THE USER AT 19 GRAZING ANGLE
C POINTS.  LINEAR INTERPOLATION IS USED BETWEEN POINTS.  THE FIRST POINT SHOULD
C BE ZERO DEGREES GRAZING ANGLE, SO THE INTERPOLATION ROUTINE CAN TREAT SMALL
C GRAZING ANGLES.
C
C     NOTE ON SOUND SPEED PROFILE DATA:  THE SOUND SPEED PROFILE DATA
C MUST EXTEND ABOVE THE SURFACE. OTHERWISE A RAY SEGMENT INTERSECTING
C THE SURFACE WILL NOT BE CALCULATED CORRECTLY.  SPECIFICALLY, THE FIRST
C THREE DATA BELOW THE SURFACE MUST BE REFLECTED IN THE SURFACE, SO THE
C PROFILE IS SYMMETRIC ABOUT THE SURFACE.  WITHOUT THIS PLOY, THE SUB-
C ROUTINE FITSSP WOULD EXTRAPOLATE THE FUNCTION BELOW THE SURFACE TO THE
C REGION ABOVE THE SURFACE, RESULTING IN THE WRONG CURVATURE FOR THE RAY
C SEGMENT IN THIS REGION.
C
C     NOTE ON SAMPLING DENSITY FOR ENVIRONMENTAL DATA:
C THE SOUND SPEED PROFILE AND BOTTOM TOPOGRAPHY ARE FIT USING PIECE-
C WISE CONTINUOUS CUBICS TO FIT FOUR POINTS AT A TIME.  HENCE ANY SHARP
C BENDS IN EITHER FUNCTION MUST BE FIT WITH AT LEAST FOUR POINTS TO KEEP
C THE POLYNOMIAL CLOSE TO THE ORIGINAL CURVE.  FAILURE TO DO SO COULD
C CAUSE THE POLYNOMIAL TO OVERSHOOT IN REGIONS WHERE IT'S NOT PINNED,
C AND LEAD TO LARGE DISCONTINUITIES IN THE DERIVATIVES WHERE SEPARATE
C SEGMENTS MEET.  THE DMPSSP AND DMPBOT ROUTINES ARE USEFUL FOR DUMPING THE
C RESULTS OF THE CURVE FITTING ROUTINES FOR LATER EXAMINATION.
C
C     AN INDEPENDENTLY COMPILED PROGRAM CALLED PLOTS (NOT A SUBROUTINE)
C IS USED TO CREATE PLOTS OF THE DATA STORED BY THIS PROGRAM.  IT IS
C SELF-DIRECTED AND REQUIRES NO INSTRUCTIONS TO RUN;  EVERYTHING IT
C NEEDS, INCLUDING INSTRUCTIONS, IS READ FROM THE FILE GENERATED BY
C THIS PROGRAM.  IF PLOTTING DATA ARE TO BE STORED, THE USER MUST HAVE
C ASSIGNED A FILE FOR THIS PURPOSE PREVIOUS TO THE RUN, AND ATTACHED
C THE UNIT NUMBER 9 TO IT.  THIS CODE, IN OTHER WORDS, EXPECTS TO BE
C ABLE TO WRITE DATA ONTO UNIT 9.  IN THE PLOTTING ROUTINE, PLOTS, THE
C EXTERNAL UNIT IS CALLED 10, AND THE USER MUST ATTACH THIS UNIT NUMBER
C TO THE FILE BEFORE EXECUTION OF PLOTS.
```

```
C     ALL INPUT IS READ FROM CARDS UNDER FREE-FIELD FORMAT. REQUIRED
C INPUT IS, BY CARD NUMBER,
C      1.  NUMBER OF SSP DATA,  AND IPLT1, A PLOT SWITCH (0 OR 1)
C      2.  SSP IN DEPTH/SPEED PAIRS, RUNNING ON FOR AS MANY CARDS AS NEEDED
C     (2A. IF IPLT1=1, THEN Z1 AND Z2 ARE NEEDED TO SPECIFY DEPTHS BOUNDS
C          IN THE SSP DUMP.)
C      3.  NUMBER OF BOTTOM TOPOG DATA,  AND IPLT2, A PLOT SWITCH (0 OR 1)
C      4.  BOTTOM TOPOGRAPHY IN RANGE/DEPTH PAIRS
C      5.  BOTTOM LOSS IN BOTLOS/BOTANG PAIRS, WHERE BOTLOS IS IN DB/BOUNCE
C          AND BOTANG IS THE GRAZING ANGLE IS DEGREES.
C      6.  RANGE STEP, SOURCE DEPTH, RECEIVER DEPTH, RECEIVER RANGE, INITIAL
C          RAY ANGLE, ANGULAR INCREMENT, AND FINAL ANGLE (ALL DEGREES), AND
C          A TOLERANCE IN THE CALCULATED RANGE TO THE BOTTOM BOUNCE POINTS.
C      7.  NUMBER OF RAY WHOSE TRAJECTORY IS TO BE PRINTED (OR ZERO),
C          FOLLOWED BY IPLT3, A SWITCH FOR PLOTTING RAY PATHS.
C     (7A. IF IPLT3=1, THEN THE NUMBER OF PATHS TO BE DRAWN IS NEEDED,
C          FOLLOWED BY A CARD(S) WITH THE INDICES OF THE RAYS, CALLING
C          RAY 1 THE FIRST RAY CALCULATED, WITH SOURCE ANGLE THETA1.)
C
C     NOTE ON UNITS:  THE CODE USES WHATEVER UNITS THE USER CHOOSES.
C THE USER MUST BE CAREFUL TO USE THE SAME LENGTH UNITS FOR THE
C SSP AND THE TOPOGRAPHIC DATA (M/S AND M, OR F/S AND F, ETC.).
C THE CHOICE OF UNITS IS IMMATERIAL SINCE THE OUTPUT INVOLVES THE SAME UNITS.
C
C A LIST OF SOME OF THE VARIABLES USED:
C
C        INPUT VARIABLES:
C          NPTS1      NO. OF SOUND SPEED POINTS
C          IPLT1      PLOTTING SWITCH FOR THE SSP
C          ZZ,CC      ARRAYS OF DEPTH, SOUND SPEED
C          Z1,Z2      DEPTHS FOR SSP DUMP IF ILPT1=1
C          NPTS2      NO. OF BOTTOM PROFILE POINTS
C          IPLT2      PLOTTING SWITCH FOR THE BOTTOM PROFILE
C          RB,ZB      ARRAYS OF RANGE, BOTTOM DEPTH
C          BOTLOS, BOTANG    ARRAYS OF BOTTOM LOSS AND GRAZING ANGLE
C          DX         RANGE STEP
C          ZS,ZR      DEPTH OF SOURCE, RECEIVER
C          R          RANGE FROM SOURCE TO RECEIVER
C          THETA1,DTHETA, THETA2  INITIAL, INCREMENT, AND FINAL ANGLES
C          DXBOT      TOLERANCE IN CALCULATING BOTTOM BOUNCE POINTS
C          ITRACE     THE INDEX OF A RAY WHOSE TRAJECTORY IS TO BE PRINTED,
                      FOR DIAGNOSTIC PURPOSES
```

```
C           IPLT3     SWITCH FOR PLOTTING OF SEVERAL RAY PATHS
C           NPATHS, IPATH  NUMBER OF PATHS TO BE PLOTTED, AND AN ARRAY OF
C                          THEIR INDECES
C
C       CALCULATED VARIABLES:
C           NRAYS     NO. OF RAYS TO BE TRACED
C           Z,DZDX,T  ARRAYS FOR STORING CURRENT VALUES OF DEPTH, SLOPE,
C                     AND TIME
C           IRAY      INDEX OF THE CURRENT RAY (1,2,...,NRAYS)
C           ZT,DZDXT,TT  PROVISIONAL VALUES OF NEW DEPTH, SLOPE, AND TIME
C           ZM1,ZM2   LAST TWO DEPTHS BEFORE CURRENT DEPTH
C           ATTEN     ARRAY OF BOTTOM LOSS ATTENUATION
C           XN,ZN,DZDXN,TN  NEW VALUES OF RANGE, DEPTH, SLOPE, AND TIME
C           ZLOC,XLOC ARRAYS FOR FITTING A CUBIC TO LOCAL RAY PATH
C           SIGN      A SWITCH (+1,-1) USED IN ITERATING TO FIND REFLECTION POINT
C           ZRT,ZBT   RAY AND BOTTOM DEPTH AT TRIAL REFLECTION POINT
C           XT        RANGE OF BOTTOM INTERSECTION OF LOCAL RAY SEGMENT
C           ANGR,ANGB ANGLE IN RADIANS OF RAY AND BOTTOM AT REFLECTION POINT
C           NHIT      NUMBER OF RAY PAIRS BRACKETING RECEIVER
C           IR,TR     ARRAY OF THE LOWER INDEX OF EACH PAIR OF RAYS BRACKETTING
C                     RECEIVER AND ARRIVAL TIME (AVG)
C           THETAS(IRAY) ANGLE OF CURRENT RAY AT SOURCE
C           THETAR(IRAY) ANGLE OF CURRENT RAY AT RCV
C           TL        ARRAY OF TRANSMISSION LOSS VALUES FOR EACH RAY-PATH
C
C
C REFS. 1.HANDBOOK OF MATHEMATICAL FUNCTIONS, ED. M.ABRAMOWITZ AND I.STEGUN,
C          NATIONAL BUREAU OF STANDARDS, NINTH PRINTING (1970), P.897.
C        2.PHYSICS OF SOUND IN THE SEA, NATIONAL DEFENSE RESEARCH COMMITTEE,
C          DEPARTMENT OF THE NAVY, VOL.8 (1949), P.53.
C
C     RICHARD C. SHOCKLEY, CODE 7213, NAVAL OCEAN SYSTEMS CENTER
C     SAN DIEGO, CALIFORNIA, 92152
C     MAY 4, 1982
C
C *******************************************************
C
C *** ASSIGN STORAGE ***
```

```
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      REAL XX,YY
      DOUBLE PRECISION K1,K2,K3,K4
      DIMENSION ZZ(200),C(200),RB(100),ZB(100),Z(400),DZDX(400),T(400),
     *          ATTEN(400),XLOC(4),ZLOC(4),TR(100),IR(100),TL(100),
     *          BOTLOS(19),BOTANG(19),THETAS(400),THETAR(400),IPATH(100)
C
      PI=3.141592653589793D00
C ********************************************************************
C *                                                                  *
C *  READ IN DATA, FIT CURVES TO THE SSP AND BOTTOM TOPOGRAPHY, AND  *
C *  PLOT THE SSP AND/OR BOTTOM TOPOGRAPHY IF DESIRED.               *
C *                                                                  *
C ********************************************************************
C
C *** GET SSP ***
C
      READ(8,1000)NPTS1,IPLT1
      READ(8,1000)(ZZ(I),C(I),I=1,NPTS1)
      IF (IPLT1.EQ.1) READ(8,1000)Z1,Z2
C
C *** GET BOTTOM TOPOG. AND BOTTOM LOSS ***
C
      READ(8,1000)NPTS2,IPLT2
      READ(8,1000)(RB(I),ZB(I),I=1,NPTS2)
      READ(8,1000)(BOTLOS(I),BOTANG(I),I=1,19)
C
C *** WRITE ENVIRONMENTAL DATA ***
C
      WRITE(6,1001)NPTS1,IPLT1
      WRITE(6,1002)(ZZ(I),C(I),I=1,NPTS1)
      WRITE(6,1014)Z1,Z2
      WRITE(6,1003)NPTS2,IPLT2
      WRITE(6,1004)(RB(I),ZB(I),I=1,NPTS2)
      WRITE(6,1010)
      WRITE(6,1011)(BOTANG(I),BOTLOS(I),I=1,19)
C
C *** FIT CURVES TO SSP AND BOTTOM ***
C
      CALL FITSSP(C,ZZ,NPTS1,0.D00)
      CALL FITBOT(ZB,RB,NPTS2,0.D00)
```

```
C *** READ PARAMETERS FOR PRESENT CASE ***
C
      READ(5,1000)DX,ZS,ZR,R,THETA1,DTHETA,THETA2,DXBOT
      WRITE(6,1005)DX,ZS,ZR,R,THETA1,DTHETA,THETA2,DXBOT
C
C
C *** READ PARAMETERS FOR DIAGNOSTIC AND RAY PATH PLOTS ***
C
      READ(8,1000)ITRACE,IPLT3
      IF (IPLT3.EQ.1)THEN
                     READ(8,1000)NPATHS
                     READ(8,1000)(IPATH(I),I=1,NPATHS)
                     WRITE(6,1015)NPATHS,(IPATH(I),I=1,NPATHS)
      END IF
C
C *** STORE SSP OR BOTTOM TOPOGRAPHY IF DESIRED ***
C
      WRITE(9,1000)IPLT1,IPLT2,IPLT3
      NPTS=R/DX + 1
      IF (IPLT3.EQ.1) WRITE(9,1000)NPATHS,NPTS
      IF (IPLT1.EQ.1)CALL DMPSSP(21,22)
      IF (IPLT2.EQ.1 .OR. IPLT3.EQ.1)CALL DMPBOT(R)
C
C
C ***************************************************************
C *                                                             *
C *   CALCULATE THE NUMBER OF RAYS TO BE TRACED AND BOTTOM ANGLE AT *
C *   THE SOURCE.  INITIALIZE THE RAY PARAMETERS.               *
C *                                                             *
C ***************************************************************
C
      CALL GETBOT(0.D0,ZS1,DZBDX)
      ANG=ATAN(DZBDX)*180./PI
      NRAYS=(THETA2-THETA1)/DTHETA +1
      WRITE(6,1006)ANG,ZS1,NRAYS
C
      DO 20 IRAY=1,NRAYS
         Z(IRAY)=ZS
         THETAS(IRAY)=THETA1+DTHETA*FLOAT(IRAY-1)
         DZDX(IRAY)=TAN((PI/180)*THETAS(IRAY))
         ATTEN(IRAY)=0
   20    T(IRAY)=0.
```

```
C ***********************************************************************
C *                                                                     *
C *   BEGIN MAIN PART. EACH RAY IS TRACED SEPARATELY.  FIRST PROVISIONAL *
C *   VALUES ARE CALCULATED, THEN SURFACE REFLECTIONS ARE CONSIDERED,AND *
C *   LAST BOTTOM REFLECTIONS.  IF A SURFACE REFLECTION OCCURS, THE RAY  *
C *   ENDPOINT AND SLOPE ARE SIMPLY INVERTED.  FOR BOTTOM REFLECTIONS,   *
C *   THE REFLECTION POINT IS FOUND ITERATIVELY, AND THE RAY SLOPE AT    *
C *   THAT POINT REDEFINED BY SPECULAR REFLECTION, FOLLOWED BY A SECOND  *
C *   RUNGE-KUTTA APPLICATION TO COMPLETE THE RANGE STEP.                *
C *                                                                     *
C ***********************************************************************
C
      DO 200 IRAY=1,NRAYS
C
C     *** DEFINE TEMPORARY VARIABLES FOR THE LAST TWO DEPTHS ***
C
         ZM1=ZS
         ZM2=ZS
C
C     *** SET A FLAG IF THE CURRENT RAY IS TO BE SAVED FOR PLOTTING ***
C
         ISAVE=0
         DO 30 I=1,NPATHS
   30    IF (IRAY.EQ.IPATH(I)) ISAVE=1
C
      IF (ISAVE.EQ.1) THEN
          XX=0
          YY=ZS
          WRITE(9,1000)XX,YY
      END IF
C
C        **********************************
C        *                                *
C        *    ENTER LOOP OVER RANGE       *
C        *                                *
C        **********************************
C
```

```
      DO 200 IX=1,INT(R/DX)
C
         XN=IX*DX
C
C        *** STEP FORWARD BY DX AND GET PROVISIONAL NEW VALUES ***
C
            DEL=DX*DZDX(IRAY)
            DZDXT=DZDX(IRAY)
            ZT=Z(IRAY)
            TT=T(IRAY)
C
            K1=DX*F(ZT,DZDXT)
            K2=DX*F(ZT+DEL/2+DX*K1/8.,DZDXT+K1/2.)
            K3=DX*F(ZT+DEL/2+DX*K1/8.,DZDXT+K2/2.)
            K4=DX*F(ZT+DEL+DX*K3/2.,DZDXT+K3)
C
            ZN=ZT+DX*(DZDXT+(K1+K2+K3)/6.)
            DZDXN=DZDXT+(K1+2.*K2+2.*K3+K4)/6.
            IF (IRAY.EQ.ITRACE)PRINT*,'XN=',XN,'ZN=',ZN,'DZDXN=',
     *                    DZDXN
            CALL GETSSP(0.5*(ZT+ZN),CAVG,DCDX)
            TN=TT+SQRT(DX*DX+(ZN-ZT)**2)/CAVG
C
C        *** CHECK FOR SURFACE REFLECTION ***
C
         IF (ZN.GT. 0) GO TO 50
C
C        *** IF THE RAY HAS HIT THE SURFACE,REFLECT IT ***
C
            ZM2=ZM1
            ZM1=Z(IRAY)
            Z(IRAY)=-ZN
            DZDX(IRAY)=-DZDXN
            T(IRAY)=TN
C
            IF (ISAVE.EQ.1)THEN
                          XX=XN
                          YY=Z(IRAY)
                          WRITE(9,1000)XX,YY
            END IF
C
            GO TO 200
```

```
   50          CONTINUE
C
C              *** CHECK FOR BOTTOM REFLECTION ***
C
               CALL GETBOT(XN,ZBOT,DZDXB)
               IF(IRAY.EQ.ITRACE)PRINT*,'ZBOT=',ZBOT,'DZDXB=',DZDXB
               IF (ZN.GT.ZBOT) GO TO 100
C
C              *** IF NO REFLECTIONS OCCURRED, USE THE PROVISIONAL ***
C                  VALUES AS THE NEW VALUES
C
               ZM2=ZM1
               ZM1=Z(IRAY)
               Z(IRAY)=ZN
               DZDX(IRAY)=DZDXN
               T(IRAY)=TN
C
               IF (ISAVE.EQ.1)THEN
                         XX=XN
                         YY=ZN
                         WRITE(9,1000)XX,YY
               END IF
C
               GO TO 200
C
  100          CONTINUE
C
C              ***   IF THE RAY HAS HIT THE BOTTOM, FIND POINT OF ***
C                    REFLECTION AND APPLY SPECULAR REFLECTION.
C                    FIRST FIT A CUBIC TO THE LOCAL RAY PATH
C
                  XLOC(1)=XN-3.*DX
                  XLOC(2)=XN-2.*DX
                  XLOC(3)=XN-DX
                  XLOC(4)=XN
                  ZLOC(1)=ZM2
                  ZLOC(2)=ZM1
                  ZLOC(3)=ZT
                  ZLOC(4)=ZN
                  CALL FITRAY(ZLOC,XLOC)
```

```
C          ***  ITERATE TO FIND POINT OF INTERSECTION OF   ***
C               RAY AND BOTTOM. XT IS THE LAST RANGE VALUE,
C               INITIALLY, AND IS USED FOR THE TRIAL REFL.PT
C
                SIGN=1
                DEL=DX/10.
                XT=XN-DX
C
  120           XT=XT+DEL
                CALL GETBOT(XT,ZBT,DZDX1)
                CALL GETRAY(XT,ZRT,DZDX2)
                DIF=ZBT-ZRT
                IF (DIF.EQ.0)GO TO 140
                IF(IRAY.EQ.ITRACE)PRINT*,'XT=',XT,'ZBT=',ZBT,
     *                'ZRT=',ZRT,'SIGN=',SIGN
                IF (SIGN*DIF.GT.0)GO TO 120
C
                SIGN=-SIGN
                DEL=-DEL/2
                IF (ABS(DEL).LT.DXBOT)GO TO 140
                GO TO 120
  140           CONTINUE
C
C          *** APPLY SPECULAR REFLECTION AT INTERSECTION POINT. ***
C
                ANGB=ATAN(DZDX1)
                ANGR=ATAN(DZDX2)
                DZDXT=TAN(2*ANGB-ANGR)

                IF(IRAY.EQ.ITRACE)PRINT*,'ANGB=',ANGB,'ANGR=',ANGR,
     *                'DZDXT=',DZDXT
C
C
C          *** CALCULATE DEPTH AND TIME AT REFLECTION POINT ***
C
                ZT=0.5*(ZBT+ZRT)
                CALL GETSSP(ZT,CAVG,DCDZ)
                TT=TT+SQRT((XT-(IX-1)*DX)**2+(ZT-Z(IRAY))**2)/CAVG
C
C          ***CALCULATE BOTTOM-LOSS ATTENUATION ***
```

```
            ANG=(180/PI)*(ANGR-ANGB)
            IF (ANG.LE.0D00 .OR. ANG.GE.90.D00) THEN
                 PRINT *,' ANG=',ANG,'IRAY=',IRAY,'XN=',XN
                 PRINT *,' ANGB=',ANGB,'DZDX1=',DZDX1
                 PRINT *,' ANGR=',ANGR,'DZDX2=',DZDX2
                 PRINT *,' ZT=',ZT,'XT=',XT
            END IF
            CALL GETBL(BOTLOS,BOTANG,ANG,BL)
            ATTEN(IRAY) = ATTEN(IRAY) + BL
C
C       *** ADVANCE THE REST OF THE RANGE STEP ***
C
            DELX=XN-XT
            DEL=DELX*DZDXT
            K1=DELX*F(ZT,DZDXT)
            K2=DELX*F(ZT+DEL/2.+DELX*K1/2.,DZDXT+K1/2.)
            K3=DELX*F(ZT+DEL/2.+DELX*K1/2.,DZDXT+K2/2.)
            K4=DELX*F(ZT+DEL+DELX*K3/2.,DZDXT+K3)
C
            Z(IRAY)=ZT+DELX*(DZDXT+(K1+K2+K3)/6.)
            DZDX(IRAY)=DZDXT + (K1+2.*K2+2.*K3+K4)/6.
            IF(IRAY.EQ.ITRACE)PRINT*,'Z(IRAY)=',Z(IRAY),
                   'DZDX(IRAY)=',DZDX(IRAY)
            CALL GETSSP(0.5*(ZT+Z(IRAY)),CAVG,DCDZ)
            T(IRAY)=TT+SQRT(DELX**2+(Z(IRAY)-ZT)**2)/CAVG
C
            IF (ISAVE.EQ.1)THEN
                          XX=XN
                          YY=Z(IRAY)
                          WRITE(9,1000)XX,YY
            END IF
C
C
C       *** END OF BOTTOM REFLECTION CALCULATION ***
C
  200 CONTINUE
C
C *******************************
C *                             *
C *   END OF RAY-TRACING PART.  *
C *                             *
C *******************************
C
C *** STORE THE ENDPOINT ANGLES ***
```

```
      DO 250 I=1,NRAYS
  250 THETAR(I)=(180/PI)*ATAN(DZDX(I))
C
C ****************************************************************
C *                                                              *
C *   BEGIN CALCULATIONS OF TRAVEL TIME AND GEOMETRICAL SPREADING *
C *   TRANSMISSION LOSS.                                         *
C *                                                              *
C ****************************************************************
C
C          *** FIRST FIND PAIRS OF RAYS ADJACENT AT THE SOURCE THAT ***
C              BRACKET THE RECEIVER.  NHIT COUNTS THE NUMBER FOUND.
C
      NHIT=0
C
      DO 350 IRAY=1,NRAYS-1
C
          IF (ZR.GE.Z(IRAY).AND.ZR.LE.Z(IRAY+1))GO TO 320
          IF (ZR.GE.Z(IRAY+1).AND. ZR.LE.Z(IRAY))GO TO 320
          GO TO 350
C
  320     CONTINUE
C
C          *** IF A RAY PAIR BRACKETS THE RECEIVER, FIND THE AVERAGE ***
C              ARRIVAL TIME AND TRANSMISSION LOSS. STORE RAY NUMBER.
C
          NHIT=NHIT+1
          TR(NHIT)=0.5*(T(IRAY)+T(IRAY+1))
          IR(NHIT)=IRAY
C
          DZ=ABS(Z(IRAY)-Z(IRAY+1))
          ANGS=0.5*(PI/180)*(THETAS(IRAY)+THETAS(IRAY+1))
          ANGR=0.5*(PI/180)*(THETAR(IRAY)+THETAR(IRAY+1))
          RATIO=(1/R)*(COS(ANGS)/COS(ANGR))*(PI/180)*DTHETA/DZ
          TL(NHIT)=-10.*ALOG10(RATIO)-10.*ALOG10(0.5*(10.**(
     *              -ATTEN(IRAY)/10)+10.**(-ATTEN(IRAY+1)/10)))
C
  350 CONTINUE
C
C          *** IF NO RAYS BRACKET RECEIVER, STOP ***
C
      IF (NHIT.GT.0) GO TO 360
      PRINT *,' NO RAY PAIRS FOUND TO BRACKET RECEIVER.'
      GO TO 450
  360 CONTINUE
C
C *** END OF TRAVEL TIME AND TRANSMISSION LOSS CALCULATIONS. ***
C      WRITE RESULTS.
C
      WRITE(6,1008)
      DO 400 I=1,NHIT
      IRAY=IR(I)
      IRP1=IRAY+1
      SPD=R/TR(I)
  400 WRITE(6,1009)IR(I),IRP1,THETAS(IRAY),THETAS(IRP1),THETAR(IRAY),
     *                THETAR(IRP1),TR(I),SPD,TL(I),Z(IRAY),Z(IRP1),

     *                ATTEN(IRAY),ATTEN(IRP1)
C
C *** PRINT SUMMARY OF RAY ENDPOINTS ***
C
  450 WRITE(6,1012)
      WRITE(6,1013)(I,Z(I),THETAR(I),T(I),I=1,NRAYS)
C
      STOP
C
 1000 FORMAT( )
 1001 FORMAT(1H1,50X,'***  IMPULSE  ***'///10X,'INPUT DATA:'///
     *        2X,'NUMBER OF PHASE VELOCITY DATA PAIRS=',I3,2X,
     *         'IPLT1=',I2//2X,'DEPTH     PHASE VEL.')
 1002 FORMAT((1H ,6(2F6.1)))
 1003 FORMAT(1H0,1X,'NO. OF BOTTOM PROFILE DATA:',I3,2X,'IPLT2=',I2,
     *        /5X,'RANGE',2X,'DEPTH')
 1004 FORMAT((1H ,6(2F6.0)))
 1005 FORMAT(1H0,4X,'RANGE STEP:',F12.1/5X,'SOURCE DEPTH:',F10.1/5X,
     *         'RECEIVER DEPTH:',F8.1/5X,'RANGE:',F17.1/5X,'THETA1:',
     *        F10.1/5X,'DTHETA:',F12.7/5X,'THETA2:',F10.1/5X,'DXBOT:',
     *         F11.1)
 1006 FORMAT(1H0,2X,'PRELIMINARY CALCULATIONS:'/5X,
     *        'BOTTOM SLOPE (DEG.) AT SOURCE:',F5.2/5X,
     *        'DEPTH AT SOURCE:',F10.1/5X,
     *        'NUMBER OF RAYS TO BE TRACED:',I4/1H1,
     *        10X,'RESULTS:'///)
```

```
1008 FORMAT(' EIGENRAYS'///5X,'RAY PAIR',5X,
    *       'SOURCE ANGS',5X,'  RCV ANGS',5X,'TIME',3X,'SIG. SP.',
    *       3X,'TRANS. LOSS(DB)',5X,'RAY-PAIR DEPTHS',5X,
    *       'BOTTOM LOSSES')
1009 FORMAT(1H ,2I5,5X,2F6.1,3X,2F6.1,2X,F6.1,4X,F6.1,7X,F10.1,6X,
    *       2F10.1,2X,2F8.1)
1010 FORMAT(1H0,1X,'BOTTOM LOSS FUNCTION'/5X,'ANG',5X,'LOSS(DB)')
1011 FORMAT((1H ,8(2F6.1)))
1012 FORMAT(1H0,'SUMMARY OF ENDPOINTS, (RAY NO., DEPTH, ANG, TIME)')
1013 FORMAT((1H ,5('(',I3,1X,F5.0,F6.2,F6.1,')')))
1014 FORMAT(1H /2X,'Z1 AND Z2(BOUNDS FOR SSP PLOT):',2F8.1)
1015 FORMAT(1H0,1X,'NO OF PATHS TO BE STORED=',I3/2X,'PATHS:',
    *       (5(20I4/)))
     END
```

## Subroutine — FITSSP

```
      SUBROUTINE FITSSP(S,Z,N,DCDZBT)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(70),B(70),C(70),D(70),S(200),Z(200),ZFOT(70)
      REAL H1,H2,H3,H1SQ,H2SQ,H3SQ,H
C
C     THIS ROUTINE FITS A SET OF PIECEWISE CONTINUOUS CUBIC POLYNOMIAL
C SEGMENTS TO AN EXISTING SET OF SOUND SPEED VALUES AND DEPTHS, S(I) AND
C Z(I),RESPECTIVELY. THE FORM OF THE POLYNOMIAL IN THE LTH LAYER IS
C
C           S(Z)=A(L)+B(L)*(Z-ZL)+C(L)*(Z-ZL)**2+D(L)*(Z-ZL)**3,
C
C WHERE ZL=Z(3*L-2) IS THE VALUE OF Z AT THE TOP OF THE LTH LAYER. WE
C FIT FOUR POINTS AT A TIME IN OTHER WORDS AND HAVE ONE POINT IN COMMON
C AT THE BOUNDARY OF ADJACENT LAYERS.  POINTS BELOW THE LAST COMPLETE
C LAYER ARE FIT SIMPLY BY LINEAR SEGMENTS, AND THE GRADIENT IN THE REGION
C BELOW THE LAST DATUM IS A USER SPECIFIED VALUE DCDZBT. N IS THE NUMBER
C OF DATA PAIRS.  THAT IS,S(I) AND Z(I) FOR I=1,2,...,N MUST BE DEFINED
C BY THE USER BEFORE CALLING FITSSP.
C N MUST BE GREATER THAN OR EQUAL TO 4 AND LESS THAN 200.
C R.C.SHOCKLEY, APRIL 1981.
C
C     CHECK FOR NUMBER OF LAYERS
C
      LAYERS=(N-1)/3
      IF (LAYERS.GE.1 .AND. N.LE.200)GO TO 20
      PRINT 101,N
  101 FORMAT(1H1,5X,'N=',I3,3X,'TOO FEW OR TOO MANY DATA FOR FITSSP')
      RETURN
C
C     BEGIN LOOP THROUGH LAYERS
C
   20 DO 40 L=1,LAYERS
         IL=3*L-2
         H1=Z(IL+1)-Z(IL)
         H2=Z(IL+2)-Z(IL)
         H3=Z(IL+3)-Z(IL)
         DS1=S(IL+1)-S(IL)
         DS2=S(IL+2)-S(IL)
         DS3=S(IL+3)-S(IL)
C
         DET=H1*H2*H3*(H2*H3*(H3-H2)+H1*H3*(H1-H3)+H2*H1*(H2-H1))
         A(L)=S(IL)
         H1SQ=H1*H1
         H2SQ=H2*H2
         H3SQ=H3*H3
         B(L)=DS1*H2SQ*H3SQ*(H3-H2)+DS2*H1SQ*H3SQ*(H1-H3)+DS3*H1SQ*H2SQ
     *        *(H2-H1)
         B(L)=B(L)/DET
         C(L)=DS1*H2*H3*(H3SQ-H2SQ)+DS2*H1*H3*(H1SQ-H3SQ)+DS3*H1*H2*
     *        (H2SQ-H1SQ)
         C(L)=-C(L)/DET
         D(L)=DS1*H2*H3*(H3-H2)+DS2*H1*H3*(H1-H3)+DS3*H1*H2*(H2-H1)
         D(L)=D(L)/DET

   40 CONTINUE
C
      DO 45 I=1,LAYERS
   45 ZBOT(I)=Z(3*I+1)
C
C     FIT LINEAR PIECES TO THE REMAINING DATA. IL IS THE INDEX OF THE
C     DATA AT THE BOTTOM OF THE LAST LAYER. L IS THE NEXT LAYER (INCOM-
C     PLETE) BELOW THE LAST COMPLETE LAYER.
C
      IL=3*LAYERS+1
      L=LAYERS+1
C
      IF (IL.EQ.N)GO TO 50
      GO TO 51
C
   50 A(L)=S(N)
      B(L)=DCDZBT
      C(L)=0
      D(L)=0
      RETURN
C
   51 IF (N.EQ.IL+1)GO TO 52
      GO TO 53
   52 A(L)=S(IL)
      B(L)=(S(N)-S(IL))/(Z(N)-Z(IL))
      C(L)=0
      D(L)=0
C
      L=L+1
      A(L)=S(N)
      B(L)=DCDZBT
      C(L)=0
      D(L)=0
      RETURN
C
   53 IF (N.EQ.IL+2)GO TO 54
      GO TO 55
C
   54 A(L)=S(IL)
      B(L)=(S(IL+1)-S(IL))/(Z(IL+1)-Z(IL))
      C(L)=0
      D(L)=0
C
      L=L+1
      A(L)=S(IL+1)
      B(L)=(S(IL+2)-S(IL+1))/(Z(IL+2)-Z(IL+1))
      C(L)=0
      D(L)=0
C
      L=L+1
      A(L)=S(N)
      B(L)=DCDZBT
      C(L)=0
      D(L)=0
      RETURN
C

   55 PRINT 102
  102 FORMAT(1H1,' PROB.  IN THE LAST DATA POINTS')
      RETURN
C
       ENTRY GETSSP(Z,SS,DCDZ)
C
```

```
C      DETERMINE THE LAYER IN WHICH ZZ LIES
C
C IF ZZ IS IN THE FIRST LAYER (WHICH MAY BE ABOVE THE SURFACE),
C USE THE COEFFICIENTS IN THE FIRST LAYER.
C
       IF (ZZ.LE.ZBOT(1)) THEN
           H=ZZ-Z(1)
           SS=A(1)+B(1)*H+C(1)*H*H+D(1)*H**3
           RETURN
       END IF
C
       DO 80 I=1,LAYERS
       IF(ZZ.GT. ZBOT(I)) GO TO 80
       L=I
       GO TO 85
    80 CONTINUE
C
C  ZZ LIES BELOW THE LAST COMPLETE LAYER.  JUMP TO THE LOGIC TO FIND,
C AND USE LINEAR INTERPOLATION IN, THE RIGHT REGION.
       GO TO 90
C
    85 H=ZZ-Z(3*L-2)
       SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
       DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
       RETURN
C
C
C
    90 IL=3*LAYERS+1
       L=LAYERS+1
       IF (N.GT.IL .AND. ZZ.GT.Z(IL+1)) GO TO 92
       H=ZZ-Z(IL)
       SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
       DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
       RETURN
C
    92 L=L+1
       IF(N.GT.IL+1 .AND. ZZ.GT.Z(IL+2)) GO TO 93
       H=ZZ-Z(IL+1)
       SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
       DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
       RETURN
```

```
C
   93 L=L+1
      H=ZZ-Z(IL+2)
      SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
      DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
      RETURN
      END
      SUBROUTINE FITBOT(S,Z,N,DCDZBT)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(33),B(33),C(33),D(33),S(100),Z(100),ZBOT(33)
      REAL H1,H2,H3,H1SQ,H2SQ,H3SQ,H
C
C     THIS ROUTINE FITS A SET OF PIECEWISE CONTINUOUS CUBIC POLYNOMIAL
C SEGMENTS TO AN EXISTING SET OF DATA STORED IN THE ARRAYS S(I) AND
C Z(I),RESPECTIVELY. THE FORM OF THE POLYNOMIAL IN THE LTH LAYER IS
C
C           S(Z)=A(L)+B(L)*(Z-ZL)+C(L)*(Z-ZL)**2+D(L)*(Z-ZL)**3,
C
C WHERE ZL=Z(3*L-2) IS THE VALUE OF Z AT THE TOP OF THE LTH LAYER. WE
C FIT FOUR POINTS AT A TIME IN OTHER WORDS AND HAVE ONE POINT IN COMMON
C AT THE BOUNDARY OF ADJACENT LAYERS.  POINTS BELOW THE LAST COMPLETE
C LAYER ARE FIT SIMPLY BY LINEAR SEGMENTS, AND THE GRADIENT IN THE REGION
C BELOW THE LAST DATUM IS A USER SPECIFIED VALUE DCDZBT. N IS THE NUMBER
C OF DATA PAIRS.  THAT IS,S(I) AND Z(I) FOR I=1,2,...,N MUST BE DEFINED
C BY THE USER BEFORE CALLING FITBOT.
C N MUST BE GREATER THAN OR EQUAL TO 4, AND LESS THAN 100.
C
C     THIS ROUTINE IS THE SAME AS FITSSP IN ACTION.  ONLY ITS NAME IS
C DIFFERENT.
C R.C.SHOCKLEY, APRIL 1981.
C
C     CHECK FOR NUMBER OF LAYERS
C
      LAYERS=(N-1)/3
      IF (LAYERS.GE.1 .AND. N.LE.100)GO TO 20
      PRINT 101,N
  101 FORMAT(1H1,5X,'N=',I3,3X,'TOO FEW OR TOO MANY DATA FOR FITBOT.')
      RETURN
C
C     BEGIN LOOP THROUGH LAYERS
C
   20 DO 40 L=1,LAYERS
        IL=3*L-2
          H1=Z(IL+1)-Z(IL)
          H2=Z(IL+2)-Z(IL)
          H3=Z(IL+3)-Z(IL)
```

```
C
      DS1=S(IL+1)-S(IL)
      DS2=S(IL+2)-S(IL)
      DS3=S(IL+3)-S(IL)
C
      DET=H1*H2*H3*(H2*H3*(H3-H2)+H1*H3*(H1-H3)+H2*H1*(H2-H1))
      A(L)=S(IL)
      H1SQ=H1*H1
      H2SQ=H2*H2
      H3SQ=H3*H3
      B(L)=DS1*H2SQ*H3SQ*(H3-H2)+DS2*H1SQ*H3SQ*(H1-H3)+DS3*H1SQ*H2SQ
     *     *(H2-H1)
      B(L)=B(L)/DET
      C(L)=DS1*H2*H3*(H3SQ-H2SQ)+DS2*H1*H3*(H1SQ-H3SQ)+DS3*H1*H2*
     *     (H2SQ-H1SQ)

      C(L)=-C(L)/DET
      D(L)=DS1*H2*H3*(H3-H2)+DS2*H1*H3*(H1-H3)+DS3*H1*H2*(H2-H1)
      D(L)=D(L)/DET
   40 CONTINUE
C
      DO 45 I=1,LAYERS
   45 ZBOT(I)=Z(3*I+1)
C
C     FIT LINEAR PIECES TO THE REMAINING DATA. IL IS THE INDEX OF THE
C     DATA AT THE BOTTOM OF THE LAST LAYER. L IS THE NEXT LAYER (INCOM-
C     PLETE) BELOW THE LAST COMPLETE LAYER.
C
      IL=3*LAYERS+1
      L=LAYERS+1
C
      IF (IL.EQ.N)GO TO 50
      GO TO 51
C
   50 A(L)=S(N)
      B(L)=DSDZBT
      C(L)=0
      D(L)=0
      RETURN
C
   51 IF (N.EQ.IL+1)GO TO 52
      GO TO 53
```

```
C
   52 A(L)=S(IL)
      B(L)=(S(N)-S(IL))/(Z(N)-Z(IL))
      C(L)=0
      D(L)=0
C
      L=L+1
      A(L)=S(N)
      B(L)=DCDZBT
      C(L)=0
      D(L)=0
      RETURN
C
   53 IF (N.EQ.IL+2)GO TO 54
      GO TO 55
C
   54 A(L)=S(IL)
      B(L)=(S(IL+1)-S(IL))/(Z(IL+1)-Z(IL))
      C(L)=0
      D(L)=0
C
      L=L+1
      A(L)=S(IL+1)
      B(L)=(S(IL+2)-S(IL+1))/(Z(IL+2)-Z(IL+1))
      C(L)=0
      D(L)=0
C
      L=L+1
      A(L)=S(N)
      B(L)=DCDZBT
      C(L)=0
      D(L)=0
      RETURN
C
   55 PRINT 102
  102 FORMAT(1H1,' PROB.  IN THE LAST DATA POINTS')
      RETURN
C
      ENTRY GETBOT(ZZ,SS,DCDZ)
```

```
C
C     DETERMINE THE LAYER IN WHICH ZZ LIES
C
      DO 80 I=1,LAYERS
      IF(ZZ.GT. ZBOT(I)) GO TO 80
      L=I
      GO TO 85
   80 CONTINUE
C
C  ZZ LIES BELOW THE LAST COMPLETE LAYER.  JUMP TO THE LOGIC TO FIND,
C AND USE LINEAR INTERPOLATION IN, THE RIGHT REGION.
      GO TO 90
C
   85 H=ZZ-Z(3*L-2)
      SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
      DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
      RETURN
C
   90 IL=3*LAYERS+1
      L=LAYERS+1
      IF (N.GT.IL .AND. ZZ.GT.Z(IL+1)) GO TO 92
      H=ZZ-Z(IL)
      SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
      DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
      RETURN
C
   92 L=L+1
      IF(N.GT.IL+1 .AND. ZZ.GT.Z(IL+2)) GO TO 93
      H=ZZ-Z(IL+1)
      SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
      DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
      RETURN
C
   93 L=L+1
      H=ZZ-Z(IL+2)
      SS=A(L)+B(L)*H+C(L)*H**2+D(L)*H**3
      DCDZ=B(L) + 2*C(L)*H + 3*D(L)*H**2
      RETURN
      END
```

Subroutine — FITRAY

```
      SUBROUTINE FITRAY(Z,X)
C
C     THIS ROUTINE FINDS THE COEFFICENTS OF A CUBIC POLYNOMIAL
C WHICH IS FIT TO FOUR DATA POINTS.  Z IS THE DEPENDENT VARIABLE, X THE
C INDEPENDENT VARIABLE, AND
C
C          Z(X)=A+B*(X-X(1))+C*(X-X(1))**2+D*(X-X(1))**3
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Z(4),X(4)
C
      H1=X(2)-X(1)
      H2=X(3)-X(1)
      H3=X(4)-X(1)
C
      DZ1=Z(2)-Z(1)
      DZ2=Z(3)-Z(1)
      DZ3=Z(4)-Z(1)
C
      DET=H1*H2*H3*(H2*H3*(H3-H2)+H1*H3*(H1-H3)+H1*H2*(H2-H1))
C
      A=Z(1)
C
      H1SQ=H1*H1
      H2SQ=H2*H2
      H3SQ=H3*H3
C
      B=DZ1*H2SQ*H3SQ*(H3-H2)+DZ2*H1SQ*H3SQ*(H1-H3)+DZ3*H1SQ*H2SQ*(H2-
     *  H1)
      C=DZ1*H2*H3*(H2SQ-H3SQ)+DZ2*H1*H3*(H3SQ-H1SQ)+DZ3*H1*H2*(H1SQ-
     *  H2SQ)
      D=DZ1*H2*H3*(H3-H2)+DZ2*H1*H3*(H1-H3)+DZ3*H1*H2*(H2-H1)
C
      B=B/DET
      C=C/DET
      D=D/DET
C
      RETURN
```

```
C
      ENTRY GETRAY(XX,ZZ,DZDX)
C
      H=XX-X(1)
      ZZ=A+B*H+C*H**2+D*H**3
      DZDX=B+2*C*H  +3*D*H**2
C
      RETURN
      END
```

Subroutine — FUNC

```
      DOUBLE PRECISION FUNCTION F(Z,DZDX)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      CALL GETSSF(Z,C,DCDZ)
      F=-(1.+DZDX**2)*(1./C)*DCDZ
C
      RETURN
      END
```

Subroutine — GETBL

```
      SUBROUTINE GETBL(Y,X,ANG,BL)
C
C  THIS ROUTINE CREATES A LINEAR PIECE-WISE CONTINUOUS FUNCTION
C  FITTING THE DATA IN ARRAYS Y AND X.  X IS REGARDED AS THE
C  INDEPENDENT VARIABLE, AND Y AS THE DEPENDENT VARIABLE.
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(20),Y(20)
C
      DO 10 I=1,18
   10 IF (ANG.GE.X(I).AND. ANG.LE.X(I+1)) GO TO 25
      PRINT *,' ERROR IN ANGLE CHECK IN GETBL. ANG=',ANG
      STOP
C
   25 BL=Y(I) + (ANG-X(I))*(Y(I+1)-Y(I))/(X(I+1)-X(I))
      RETURN
      END
```

Subroutine — PLTSSP

```
      SUBROUTINE DMPSSP(Z1,Z2)
C
      DOUBLE PRECISION Z1,Z2,C,DCDZ,Z
C
C   THIS ROUTINE GETS 300 SOUND SPEED POINTS FROM Z1 TO Z2 USING THE FIT
C GENERATED BY FITSSP AND DUMPS THE RESULT ON UNIT 9.
C
      DO 50 I=1,300
      Z=Z1+(I-1)*(Z2-Z1)/300
      CALL GETSSP(Z,C,DCDZ)
      ZZ=Z
      CC=C
   50 WRITE(9,1000)ZZ,CC
C
      RETURN
 1000 FORMAT( )
      END
```

Subroutine — PLTBOT

```
      SUBROUTINE DMPLOT(R)
C
      DOUBLE PRECISION R,X,Z,DZDX
C
C     THIS ROUTINE GETS 300 BOTTOM DEPTHS EVENLY SPACED BETWEEN SOURCE
C AND RCV AND DUMPS THE RESULT ON UNIT 9.
C
      DO 50 I=1,300
      X=(I-1)*R/300
      CALL GETBOT(X,Z,DZDX)
      XX=X
      ZZ=Z
   50 WRITE(9,1000)XX,ZZ
C
      RETURN
 1000 FORMAT( )
      END
```

```
C     THIS ROUTINE IS A PLOTTING ADJUNCT TO IMPULSE.  IT USES A FILE
C IN WHICH ARE STORED BOTH PLOTTING DATA AND INSTRUCTIONS.  THE STRUCTURE
C OF THE FILE IS:
C             LINE 1 IPLT1,IPLT2,IPLT3
C             IF IPLT3=1, LINE 2 HAS NPATHS, NPTS
C WHERE NPATHS IS THE NUMBER OF RAY TRAJECTORIES SAVED, AND NPTS IS THE
C NUMBER OF POINTS IN EACH TRAJECTORY.
C             IF IPLT3=0, LINE 2 BEGINS THE DATA ITSELF.
C             THE SSP COMES NEXT, IF SAVED, CONSISTING OF 300 POINTS IN
C DEPTH/SOUND SPEED PAIRS.
C             IF IPLT2=1 (OR IPLT3=1), THIS IS FOLLOWED BY 300 BOTTOM TOPO-
C GRAPHIC POINTS IN RANGE/DEPTH PAIRS.
C             IF IPLT3=1, THIS IS FOLLOWED BY A SET OF NPATHS RAY TRAJEC-
C TORIES, EACH WITH NPTS DATA IN RANGE/DEPTH PAIRS.
C     THE PLOTS GENERATED ARE AN SSP IF IPLT=1, A BOTTOM TOPOGRAPHIC
C PLOT IF IPLT2=1 AND IPLT3=0, AND A RAY-PATH PLOT TOGETHER WITH BOTTOM
C TOPOGRAPHY IF IPLT3=1.
C
      DIMENSION X(3000),Y(3000)
C
      READ(10,1000)IPLT1,IPLT2,IPLT3
      PRINT *,'IPLT1=',IPLT1,', IPLT2=',IPLT2,', IPLT3=',IPLT3
C
      IF (IPLT3.EQ.1)THEN
                        READ(10,1000)NPATHS,NPTS
                        PRINT *,'NPATHS=',NPATHS,', NPTS=',NPTS
      END IF
C
C *** FIRST MAKE SSP PLOT IF DESIRED ***
C
      IF (IPLT1.EQ.1) THEN
            DO 10 I=1,300
   10       READ(10,1000)X(I),Y(I)
            CALL LINPLT(X,Y,300)
      END IF
C
C *** NOW PLOT BOTTOM DEPTH AND RAY PATHS IF IPLT3=1.  OTHERWISE, IF
C     IPLT2=1, JUST MAKE PLOT OF BOTTOM DEPTH.
```

```
      IF (IPLT2.EQ.1 .AND. IPLT3.EQ.0)THEN
      PRINT *,' BOTTOM DEPTH DATA TO BE PLOTTED:'
            DO 20 I=1,300
   20       READ(10,1000)X(I),Y(I)
            CALL LINPLT(X,Y,300)
           STOP
      END IF
C
C
      IF (IPLT3.EQ.1)THEN
            DO 30 I=1,300
   30       READ(10,1000)X(I),Y(I)
            R=X(300)
C
              IN=INT(ALOG10(R))
              RATIO=R/10**IN
              ROUND=INT(RATIO+0.5)
              XSTP=ROUND*10**(IN-1)
C
C *** FIND MAX DEPTH OF BOTTOM AND USE IT TO CALCULATE THE DEPTH
C     LIMIT FOR PLOTS OF RAY PATHS AND BOTTOM PROFILE, AND THE
C     DEPTH STEP SIZE.
C
              YMAX=Y(1)
              DO 35 I=2,300
                   IF (Y(I) .GT. YMAX) YMAX=Y(I)
   35         CONTINUE
C
              IN=INT(ALOG10(YMAX))
              REM=ALOG10(YMAX) - IN
              DMAX=INT(10**REM +1)*(10**IN)
              DSTEP=-DMAX/5.
C
C
             CALL BGNPL(1)
             CALL TITLE('RAY PATHS$',-100,'RANGE$',100,'DEPTH$',100,
     *                  0.,6.)
             CALL GRAF(0.,XSTP,R,DMAX,DSTEP,0.)
C
             CALL CURVE(X,Y,300,0)
```

```
              DO 50 I=1,NPATHS
                   DO 40 J=1,NPTS
   40              READ(10,1000)X(J),Y(J)
C
C
   50         CALL CURVE(X,Y,NPTS,0)
C
              CALL ENDPL(1)
         END IF
C
 1000    FORMAT( )
C
         STOP
         END
```

# APPENDIX B. LISTING OF TEST PROGRAMS

The two test routines presented here create data files for IMPULSE. The first creates data for a Hirsch profile; the second creates data for a parabolic mirror bottom.

```
C      TEST:  THIS CODE CREATES A DATA SET SUITABLE FOR TEST CASE RUNS
C    OF IMPULSE, AND STORES THE DATA ON A FILE (EXTERNAL UNIT 9), USING
C    THE HIRSCH PROFILE:
C
C       C(Z) = C0/SQRT(1-(Z-Z0)**2/A**2).
C
C       THE MEANING OF MOST VARIABLES IN THIS CODE IS THE SAME AS
C    THAT IN IMPULSE.  THE SOURCE AND RECEIVER ARE ON THE AXIS.
C    THE BOTTOM IS AT DEPTH 2*Z0.  THE AXIS DEPTH Z0 MUST BE LESS THAN
C    A TO ENSURE THAT THE CURVE-FIT ROUTINE HAS DATA BEGINNING AT THE
C    SURFACE.  THE BOTTOM IS LOSSLESS AND FLAT.  A PLOT OF THE BOTTOM
C    ALONE GIVES DISSPLA NORMALIZATION PROBLEMS, SO BOTTOM DEPTH
C    IS ONLY PLOTTED IF THE RAYS ARE PLOTTED ALSO.
C       SURFACE REFLECTIONS ARE OF NO INTEREST, AND THEREFORE
C    THE SSP IS NOT REFLECTED IN THE SURFACE FOR THE FIRST FOUR POINTS.
C    NPTS1 IS THE NUMBER OF SSP DATA CREATED AND STORED.
C
C       NOTE: IMPULSE MUST BE MODIFIED TO USE THIS STORED DATA,
C    SINCE IT MUST READ DATA FROM AN EXTERNAL UNIT RATHER THAN CARDS.
C
C       THE DATA READ IN BY THIS CODE IS AS FOLLOWS:
C           C0, Z0, A
C           NPTS1, IPLT1
C           Z1, Z2 (IF IPLT1=1)
C           DX, R, THETA1, DTHETA, THETA2
C           IPLT3
C           NPATHS (IF IPLT3=1)
C           IPATH(1), . . ., IPATH(NPATHS)   (IF IPLT3=1)
C
      DIMENSION ZZ(200),C(200),RB(10),ZB(10),BOTANG(19),BOTLOS(19)
     *             ,IPATH(100)
C
      READ(5,1000) C0,Z0,A
      WRITE(6,1001) C0,Z0,A
C
      READ(5,1000)NPTS1,IPLT1
      WRITE(6,1002)NPTS1,IPLT1
      WRITE(9,1000)NPTS1,IPLT1
```

```
C  GENERATE SSP AND STORE IT.
C
      DO 20 I=1,NPTS1
         ZZ(I)=(I-1)*2*Z0/(NPTS1-1)
         OFF = ZZ(I)-Z0
   10    C(I)=C0/SQRT(1-(OFF/A)**2)
   20 CONTINUE
C
      WRITE(6,1003)
C
      DO 30 I=1,NPTS1
   30 WRITE(6,1004)ZZ(I),C(I)
C
      WRITE(9,1000)(ZZ(I),C(I),I=1,NPTS1)
C
C  IF SSP PLOT IS DESIRED, READ AND STORE ITS DEPTH EXTENT.
C
      IF (IPLT1.EQ.1) THEN
         READ(5,1000)Z1,Z2
         WRITE(6,1005)Z1,Z2
         WRITE(9,1000)Z1,Z2
      END IF
C
      READ(5,1000) DX,R,THETA1,DTHETA,THETA2
C
C  GENERATE BOTTOM AT DEPTH 2*Z0, USING 10 DATA, AND STORE THEM.
C
      NPTS2=10
      IPLT2=0
C
      WRITE(6,1006)NPTS2,IPLT2
      WRITE(9,1000)NPTS2,IPLT2
C
      DO 40 I=1,10
         RB(I)=(I-1)*R/9
         ZB(I)=2.*Z0
   40 CONTINUE
C
      WRITE(6,1007)
      DO 50 I=1,10
   50 WRITE(6,1008)RB(I),ZB(I)
C
      WRITE(9,1000)(RB(I),ZB(I),I=1,10)
C
C  GENERATE BOTTOM LOSS AND STORE
C
      WRITE(6,1009)
C
      DO 60 I=1,19
            BOTANG(I)=5*(I-1)
            BOTLOS(I)=0.
   60       WRITE(6,1010)BOTANG(I),BOTLOS(I)
      WRITE(9,1000)(BOTLOS(I),BOTANG(I),I=1,19)
C
      ZS=Z0
      ZR=Z0
      DXBOT=0
C
      WRITE(6,1011)DX,ZS,ZR,R,THETA1,DTHETA,THETA2,DXBOT
      WRITE(9,1000)DX,ZS,ZR,R,THETA1,DTHETA,THETA2,DXBOT
C
      ITRACE=0
      READ(5,1000)IPLT3
C
      WRITE(6,1012)ITRACE,IPLT3
      WRITE(9,1000)ITRACE,IPLT3
C
      IF (IPLT3.EQ.1) THEN
            READ(5,1000)NPATHS
            READ(5,1000) (IPATH(I),I=1,NPATHS)
            WRITE(6,1013)NPATHS
            WRITE(9,1000)NPATHS
C
            WRITE(6,1014)(IPATH(I),I=1,NPATHS)
            WRITE(9,1000) (IPATH(I),I=1,NPATHS)
      END IF
C
      STOP
```

```
1000 FORMAT( )
1001 FORMAT(1H1,5X,'C0=',E10.3,5X,'Z0=',F10.3,5X,'A=',F10.7)
1002 FORMAT(1H0,5X,'NPTS1=',I5,5X,'IPLT1=',I2)
1003 FORMAT(1H0//10X,'SSP DATA:'/)
1004 FORMAT(1H ,5X,5(2E12.4))
1005 FORMAT(1H0,5X,'Z1 AND Z2=',2E15.4)
1006 FORMAT(1H0,5X,'NPTS2=',I5,5X,'IPLT2=',I2)
1007 FORMAT(1H0/10X,'BOTTOM DEPTH:'/)
1008 FORMAT(1H ,5X,5(2E12.4))
1009 FORMAT(1H0/10X,'BOTTOM LOSS:'/)
1010 FORMAT(1H ,5X,5(2E12.4))
1011 FORMAT(1H0,'DX=',E12.5/' ZS=',E12.5/' ZR=',E12.5/' R=',E12.5/
     *        ' THETA1=',E10.4/' DTHETA=',E10.4/' THETA2=',E10.4/
     *        ' DXBOT=',E10.4)
1012 FORMAT(1H0/5X,'ITRACE=',I3,5X,'IPLT3=',I3)
1013 FORMAT(1H0,5X,'NPATHS=',I3)
1014 FORMAT(1H0,5X,'PATHS TO BE STORED:'/2X,(5(2I14)))
     END
```

## Test Case for Bottom Reflections

```
C        THIS CODE GENERATES A FILE CONTAINING DATA TO BE USED AS INPUT TO PULSE
C     FOR TESTING THE BOTTOM REFLECTION ACCURACY.  THE PHASE VELOCITY IS CONSTANT
C     SO RAYS SHOULD BE SEGMENTS OF STRAIGHT LINES.  THE BOTTOM IS A PARABOLIC
C     REFLECTOR, WITH THE SOURCE AT THE FOCUS:
C
C              ZB = ZS + SQRT(A**2+2*A*R),
C
C     WHERE ZS IS THE SOURCE DEPTH, A IS THE PARAMETER FIXING THE RATE OF INCREASE
C     OF DEPTH WITH RANGE, AND R IS RANGE.
C
C        TEN POINTS ARE USED TO SPECIFY THE SSP, AND THE SOUND SPEED IS SET TO
C     1500.  THE MEANING OF THE VARIABLES IS THE SAME AS IN PULSE.
C
C        THE INPUT DATA IS AS FOLLOWS:
C              NPTS2, A, DXBOT
C              ZS,ZR,DX,R,THETA1,DTHETA, THETA2
C              IPLT3
C              NPATHS (IF IPLT3=1)
C              IPATH(1), . . ., IPATH(NPATHS) (IF IPLT3=1)
C
C
         DIMENSION ZZ(10),C(10),RB(100),ZB(100),BOTANG(19),
        *          BOTLOS(19),IPATH(100)
C
C        READ THE NUMBER OF BOTTOM PROFILE DATA, A, AND DX.
C
         READ(5,1000)NPTS2,A,DXBOT
         WRITE(6,1001)NPTS2,A,DXBOT
C
C        READ THE OTHER PARAMETERS ZS,ZR,DX,R,THETA1,DTHETA, AND THETA2.
C
         READ(5,1000)ZS,ZR,DX,R,THETA1,DTHETA,THETA2
         WRITE(6,1002)ZS,ZR,DX,R,THETA1,DTHETA,THETA2
C
C        GENERATE THE SSP AND STORE IT.
```

```
      ZMAX=ZS + SQRT(A*A+2*A*R)
      NPTS1=10
      IPLT1=0
      DEL=ZMAX/9.
C
      WRITE(9,1000)NPTS1,IPLT1
C
      DO 20 I=1,NPTS1
           ZZ(I)=(I-1)*DEL
           C(I)=1500.
   20 CONTINUE
C
      WRITE(9,1000)(ZZ(I),C(I),I=1,NPTS1)
      WRITE(6,1003)(ZZ(I),C(I),I=1,NPTS1)
C
      IPLT2=1
      WRITE(9,1000)NPTS2,IPLT2
C
C      GENERATE THE BOTTOM PROFILE AND STORE.
C
      DO 30 I=1,NPTS2
           RB(I)=(I-1)*R/(NPTS2-1)
           ZB(I)=ZS+SQRT(A*A+2*A*RB(I))
   30 CONTINUE
C
      WRITE(6,1004)(RB(I),ZB(I),I=1,NPTS2)
      WRITE(9,1000)(RB(I),ZB(I),I=1,NPTS2)
C
C      GENERATE THE BOTTOM LOSS FUNCTION AND STORE.
C
      DO 40 I=1,19
           BOTANG(I)=5*(I-1)
           BOTLOS(I)=0.
   40 CONTINUE
C
      WRITE(6,1005)(BOTLOS(I),BOTANG(I),I=1,19)
      WRITE(9,1000)(BOTLOS(I),BOTANG(I),I=1,19)
C
C     WRITE THE REMAINING PARAMETERS ONTO THE DATA FILE.
C
      WRITE(9,1000)DX,ZS,ZR,R,THETA1,DTHETA,THETA2,DXBOT
```

```
C     IF RAY PLOTS ARE DESIRED, READ AND STORE THE INSTRUCTIONS FOR THEM.
C
      ITRACE=0
      READ(5,1000)IPLT3
C
      WRITE(6,1006)ITRACE,IPLT3
      WRITE(9,1000)ITRACE,IPLT3
C
      IF (IPLT3.EQ.1)THEN
            READ(5,1000)NPATHS
            READ(5,1000)(IPATH(I),I=1,NPATHS)
            WRITE(6,1007)NPATHS
            WRITE(9,1000)NPATHS
            WRITE(6,1008)(IPATH(I),I=1,NPATHS)
            WRITE(9,1000)(IPATH(I),I=1,NPATHS)
      END IF
C
      STOP
C
 1000 FORMAT( )
 1001 FORMAT(1H1,5X,'NPTS2=',I3,5X,'A=',E10.3,5X,'DXBOT=',E10.3)
 1002 FORMAT(1H0,5X,'ZS=',E10.3,5X,'ZR=',E10.3,5X,'DX=',E10.3,
     *       /4X,'R=',E12.5,5X,'THETA1=',E10.4,5X,'DTHETA=',E10.4/
     *       4X,'THETA2=',E10.4)
 1003 FORMAT(1H0,5X,'SSP DATA:'/(5(2F10.0,1X)))
 1004 FORMAT(1H0,5X,'BOTTOM PROFILE:'/(4(2E12.4)/))
 1005 FORMAT(1H0,5X,'BOTTOM LOSS:'/(4(2E12.4)/))
 1006 FORMAT(1H0,5X,'ITRACE=',I4,5X,'IPLT3=',I4)
 1007 FORMAT(1H0,5X,'NPATHS=',I4)
 1008 FORMAT(1H0,5X,'IPATH ARRAY:'/2X,(5(20I4)))
C
      END
```